Sisteme de operare

Litera A

ABI (Applications Binary Interface) = interfață pentru aplicații la nivel de cod

Programe care asigură compatibilitatea aplicațiilor cu sistemul de calcul la nivelul cel mai scăzut: hardware. Într-un model ierarhic al aplicațiilor, acesta este primul nivel.

abort (to) = a abandona

Terminarea forțată a unui proces (activități) datorită unei condiții de eroare sau deciziei utilizatorului (sau a administratorului). Terminarea poate fi făcută în mod asincron față de execuția procesului.

absolute loader = încărcător absolut

Modul program, componentă a unui sistem de operare, care realizează transferul unui program generat în format binar absolut (sau a unei porțiuni dintr-un program) de pe un suport de memorie externă în memoria internă. Acest tip de încărcător nu realizează şi relocarea programului ci numai încărcarea la adrese fixe din memorie, cunoscute în momentul construirii programului executabil (construcție realizată prin compilare/asamblare şi editare de legături). Acțiunea de încărcare se realizează după alocarea memoriei de către sistemul de operare şi precede execuția programului.

access (to) = a face acces

1. Accesul utilizatorului la resursele unui sistem de calcul prin intermediul sistemului de operare care îi oferă o maşină virtuală. Sistemele cu multiacces permit lucrul simultan de la mai multe terminale (fizice sau emulate, locale sau aflate la distanță) a mai multor utilizatori.
2. Accesul unui proces la o resursă ce i-a fost alocată de către sistemul de operare.
3. Accesul la un bloc de informații memorate pe un suport extern (disc, bandă, casetă, tambur magnetic). În sens mai larg informațiile organizate ca înregistrări în fişiere sunt şi ele resurse gestionate de către sistem.

access time = timp de acces

Timpul necesar accesării unei informații aflată pe suport extern. Pentru dispozitivele cu capete mobile:

tacces = t deplas + t rot + t trans

- t deplas - este timpul necesar deplasării capetelor de pe cilindrul (pista) curent pe cel destinație. Este componenta cu ponderea cea mai mare din timpul de acces. Se mai numeşte şi "timp de căutare" pentru că uneori capetele nu ajung după deplasare pe pista vizată şi mai este necesară o mişcare pentru corecție.
- t rot - numit şi "timp de latență" este timpul necesar pentru ca, prin rotirea discului, blocul dorit să ajungă sub capul de citire. În medie acest timp este egal cu jumătate din perioada de rotație a discului.
- t trans - timpul de transfer este timpul necesar transferului informației de pe disc în buffer-ul din memorie. Este o caracteristică cunoscută a discului.

Pentru dispozitivele cu capete fixe:

tacces = t rot + t trans

componentele având aceeaşi semnificație ca mai sus. Datorită absenței timpului de transfer valoarea este mult mai mică.

Ada - Ada (limbajul de programare)

Limbaj de programare proiectat pornind de la Pascal în urma evaluării unui mare număr de limbaje de programare. Ada este limbajul obligatoriu impus de Pentagon pentru proiectele software ale Departamentului Apărării a SUA. Numele nu este un acronim, ci aminteşte de primul programator din lume, Augusta Ada Byron, contesă de Lovelace, fiica lordului Byron şi

asistenta lui Charles Babbage. Limbajul este orientat spre programare modulară, are o mare elasticitate în ceea ce priveşte tipurile de date, aduce o abordare nouă pentru tratarea excepţiilor program. Mecanismele de multitasking poartă numele de *rendezvous*. Implementările pe diverse arhitecturi sunt însoţite şi de instrumente de ingineria programării. S-au făcut eforturi deosebite pentru standardizare, norma în vigoare fiind Ada 95. Proiectul deosebit de vast şi costisitor care a condus la elaborarea şi implementarea acestui limbaj (colectivul care a dezvoltat limbajul a fost condus de Jean Ichbiah), ca şi ambiţia de a realiza un limbaj adecvat oricărui tip de aplicaţii fac ca Ada să fie uneori considerat "PL/I al anilor 1980".

address space = spaţiul adreselor

Colecţia de module program şi de date pe care le adresează un proces, sau totalitatea adreselor generate de procesor când execută un proces. Acest spaţiu este construit în faza de editare de legături, punând la un loc module obiect rezultate din compilarea programelor sursă ale utilizatorului sau preluate din biblioteci, module de date iniţializate sau numai rezervate, spaţiul rezervat pentru stivă, etc. Module reentrante se pot regăsi în spaţiul de adrese al mai multor procese (componente ale sistemului de gestiune a fişierelor, rutine matematice frecvent utilizate, etc.). Spaţiul de adrese trebuie mapat pe memoria fizică la momentul execuţiei, fiind posibil ca numai o parte a lui să fie la un moment dat în memoria sistemului de calcul. Elementele fundamentale pentru construirea acestui spaţiu se găsesc memorate într-un fişier care conţine programul executabil. La UNIX segmentul de cod pur *.text* poate fi partajat de mai multe procese şi de aceea nu suferă procesul de swapping, în timp ce alte module se duplică în momentul execuţiei unui apel fork() şi pot fi evacuate. În sistemul RSX-11M spaţiul de adrese numit şi *spaţiu virtual* era limitat la 64Kb datorită registrelor de 16 biţi. *Spaţiul fizic* fiind mult mai mare erau necesare mecanisme speciale de mapare (registrele de pagină APR). Pentru a spori posibilităţile programelor s-a introdus noţiunea de *spaţiu logic* al adreselor. Prin maparea succesivă a unor ferestre virtuale pe diferite regiuni ale memoriei fizice, task-ul poate adresa în timpul execuţiei sale orice cantitate de memorie fără a depăşi însă în nici un moment limitarea impusă de dimensiunea registrelor (în orice moment spaţiul nu poate de păşi 64Kb).

administrator = administrator

Persoană însărcinată cu coordonarea şi controlul acţiunilor referitoare la un sistem de calcul (sau ansamblu de sisteme de calcul) aflate sub controlul unuia sau mai multor sisteme de operare. Un sistem de operare pune la dispoziţia administratorului următoarele categorii de servicii specializate:

- servicii pentru identificarea utilizatorilor şi asigurarea securităţii accesului în sistem
- servicii pentru asigurarea comunicării între utilizatori
- servicii pentru măsurarea performanţelor sistemului
- servicii pentru gestiunea dispozitivelor (discuri, benzi) din sistem
- servicii pentru administrarea sistemului de fişiere şi a imprimantelor din sistem

Administratorul poate stabili valorile unor parametri specifici care influenţează caracteristicile de exploatare a sistemului. El realizează şi reconfigurarea sistemului în cazul efectuării unor modificări asupra elementelor hardware/software care compun sistemul de calcul.

AFS (Andrew File System) = AFS

Sistem de fişiere distribuit dezvoltat la Carnegie Mellon University (CMU). În anii 1980 a fost dezvoltat un proiect de sistem distribuit având scopul de a oferi fiecărui utilizator (student sau membru al personalului didactic şi de cercetare) acces la o staţie de lucru care lucrează în UNIX BSD. Un număr de reţele locale, fiecare având un server şi un număr de staţii client sunt interconectate într-o reţea de dimensiuni mari, care a fost extinsă şi într-un campus al altei universităţi aflată la peste 150 km distanţă. Serverele rulează un proces special "multithread". Spaţiul numelor arată ca un arbore tradiţional UNIX, la ierarhia locală de fişiere adăugându-se un catalog */cmu* al cărui conţinut este suportat de către AFS prin intermediul serverelor. Pentru ca cea mai mare parte a traficului să se producă local, la deschiderea unui fişier aflat la distanţă întregul fişier sau o parte din el este copiat într-un "cache" aflat pe discul local. Acest lucru este transparent pentru utilizator, staţia de lucru utilizând fişierul în mod eficient, ca pe un fişier UNIX obişnuit. Numai funcţia *open()* a fost modificată faţă de un sistem BSD obişnuit. La închiderea unui fişier copiat de pe un server, copia sa este păstrată pe discul local pentru eventuale utilizări viitoare. Mecanismele care asigură coerenţa cache-urilor sunt activate numai la cererea utilizatorului, pentru a nu creşte traficul în reţea. În mod normal, când un proces deschide un fişier deja deschis de alt proces, el "vede" copia de pe discul local, care ar putea fi diferită de cea aflată pe server. Existenţa unui număr de circa 10.000 de utilizatori ridică probleme de securitate deosebite.

AIX - Advanced Interactive eXecutive = AIX

Numele dat de IBM pentru diferitele implementări ale sistemului de operare UNIX pe arhitecturi hardware produse de IBM.

allocate (to) = a aloca

Acţiunea de a asocia o resursă a unui sistem de calcul unei anumite activităţi din sistem, pentru a permite continuarea desfăşurării acelei activităţi. Alocarea resurselor reprezintă una din funcţiile fundamentale ale unui sistem de operare. Ea se poate referi la spaţiul de memorie (internă sau externă), timpul procesor, canale de comunicaţie, informaţii memorate în fişiere, echipamente periferice, etc. Alocarea poate fi statică (dacă resursele sunt puse la dispoziţia procesului indiferent dacă acesta le utilizează sau nu) sau dinamică (se face în funcţie de necesităţile de la un anumit moment).

answer time = timp de răspuns

Timpul în care este tratată complet o cerere adresată sistemului. De exemplu, pentru lucrarea *i* (job) timpul de răspuns Ti este:

Ti = TTi - TSi

unde:

TTi - este momentul terminării

TSi - este momentul sosirii lucrării în sistem

API (Application Programming Interface) = interfaţă de programare a aplicaţiilor

Parte integrantă a programelor de sistem, acest tip de interfaţă se află plasată între o componentă hardware (placă de sunet, interfaţă grafică, etc) şi programul de aplicaţie care o utilizează. Se permite dezvoltarea de aplicaţii independente de un hardware particular, furnizorul echipamentului oferind numai driver-ul final care converteşte instrucţiunile generice ale interfeţei în comenzi specifice acelui echipament. Tehnica poate fi folosită şi pentru programe de bază (de exemplu Microsoft Windows), în loc de dispozitive hardware.

append (to) = a adăuga

Operaţie de lucru cu un fişier, care constă în adăugarea de informaţii la sfârşitul fişierului, după ultima înregistrare a acestuia. La deschiderea fişierului trebuie specificat modul "append" pentru ca sistemul să asigure extinderea fişierului prin alocarea spaţiului suplimentar.

application software = programe de aplicaţii

Totalitatea programelor de aplicaţie asociate unui anumit sistem de calcul. Spre deosebire de *programele de bază*, care controlează resursele sistemului şi asigură funcţionarea sa, cele de aplicaţie au rolul de a rezolva problemele specifice de prelucrare a informaţiilor ale utilizatorilor. De exemplu sistemele de gestiune a bazelor de date, jocurile pe calculator, sistemele de proiectare asistată de calculator, foile de calcul sau sistemele de publicare asistată de calculator pot fi încadrate în această categorie. În general aceste programe fac apel la servicii oferite de către sistemul de operare.

asynchronous input/output = operaţie de intrare/ieşire asincronă

Operaţie de transfer de date care se desfăşoară asincron în raport cu execuţia programului, suprapunându-se parţial în timp cu aceasta. Sistemele de operare în timp-real oferă cu precădere apeluri sistem pentru operaţii asincrone, precum şi mecanismele prin care procesul poate testa dacă operaţia s-a încheiat, poate aştepta terminarea ei, sau este informat despre această terminare. Sistemul UNIX, sistem cu divizarea timpului dar şi cu extensii de timp-real, oferă apeluri pentru operaţii atât sincrone cât şi asincrone.

atomic = indivizibil

Operaţie care nu poate fi întreruptă. De exemplu primitivele de sincronizare **test_and_set** sunt "atomice" în sensul că

realizează operațiile de testare şi de modificare a variabilei de blocare ("zăvor") ca pe o operație unică, împiedicând astfel testarea valorii ei de către mai multe procese simultan. Foarte multe mecanisme de sincronizare se implementează ca operații indivizibile.

attach (to) = a ataşa

Prin acțiunea de ataşare a terminalului, un proces creează un canal de transfer de informații, utilizatorul putând să interacționeze cu procesul (task-ul) aflat în execuție.

auxiliary memory = memorie auxiliară

Porțiune a memoriei unui sistem de calcul care are rolul de a păstra cantități mari de informație (programe sau date) pentru a putea fi aduse în memoria internă în vederea prelucrării. Memoria auxiliară are capacități mai mari dar şi timpi de acces mai mari față de memoria internă. Costul pe bit memorat este mai mic decât în cazul memoriei interne. Memoria auxiliară păstrează informația într-un mod persistent (nevolatil) în sensul că informația continuă să rămână pe suportul extern de informație chiar după dispariția alimentării. Exemple specifice sunt memoriile auxiliare realizate pe discuri magnetice cu capete fixe sau mobile, pe discuri realizate în tehnologie Winchester, pe dischete (discuri flexibile), pe compact discuri (CD-ROM), pe tambururi magnetice sau diverse variante de memorie pe bandă magnetică. Un termen echivalent este cel de memorie externă.

average access time = timp mediu de acces

Media timpilor de acces la informații. (A se vedea şi "timp de acces")

awk = awk (limbajul)

Limbaj din lumea sistemelor de operare UNIX dezvoltat de Alfred Aho, Peter Weinberger şi Brian Kernighan. Serveşte la prelucrarea textelor (unul dintre scopurile principale pentru care a fost construit primul sistem UNIX) şi se caracterizează prin: o sintaxă înrudită cu limbajul C, prelucrare orientată pe câmpuri, lipsa declarațiilor de tip pentru variabile.

Litera B

background job = lucrare cu prioritate redusă

În sistemele de operare cu prelucrare pe loturi ("batch processing") termenul desemnează o lucrare neinteractivă căreia i se asociază de obicei (în algoritmul de planificare spre execuție) un nivel de prioritate mai scăzut. Se consideră că în sistem există două cozi: în coada din primul plan ("foreground") se află lucrările cu prioritate mare, iar în cea din fundal ("background") lucrările cu prioritate mai mică; acestea nu se planifică decât atunci când prima coadă este vidă.

background process = program executat în fundal

Se consideră cazul unui sistem de operare multiproces în care utilizatorul interacționează cu sistemul prin intermediul unui terminal. Un proces lansat în execuție de către utilizator fără ca acesta (procesul) să dețină controlul fişierului standard de intrare se numeşte "background process". El este detaşat de la terminalul de la care a fost lansat şi adesea rulează cu o prioritate mai mică. Termenul a fost pentru prima dată introdus la OS/360, iar în momentul de față este utilizat la UNIX printre ale cărui caracteristici se numără şi cele de *multiproces* şi *multiacces*.

backing store = memorie auxiliară

Memorie nevolatilă de capacitate mai mare decât memoria principală, dar cu timp de acces mai mare (de exemplu: disc magnetic, tambur magnetic, bandă magnetică). Utilizată uneori pentru realizarea de copii de salvare ale unor informații în organizări diverse. Prețul mare al memoriei interne face ca dimensiunea ei să fie limitată; memoriile auxiliare sunt mult mai ieftine şi pot avea dimensiuni mult mai mari. Prin utilizarea memoriei tampon se realizează o creştere a performanțelor.

back-up (to) = a salva

Deoarece informațiile memorate în fişiere se pot pierde (în urma unor erori de programare sau de operare, în urma unor acțiuni răuvoitoare sau în urma deteriorării suportului magnetic), periodic se fac copii de rezervă ale acestora pentru a se putea restaura fişierele pierdute. Se salvează sisteme de fişiere în întregime, părți din ele, sau fişiere individuale. Suportul pe care se realizează copia poate fi o bandă magnetică, un disc de mare capacitate, sau un mediu obişnuit, identic cu cel salvat. Uneori, informațiile salvate se compactează, astfel încât să ocupe cât mai puțin spațiu, accesul la ele urmând a se face secvențial cu ocazia restaurării şi nu pentru regăsire şi accesare rapidă. Sarcina salvării informațiilor revine administratorului de sistem, dar în sistemele de operare moderne rolul sistemului de operare este foarte mare. Astfel, în UNIX activitatea de salvare se planifică după o anumită strategie (care stabileşte ce se salvează, cu ce periodicitate şi pe ce suport magnetic pentru a refolosi după un timp vechile benzi), iar sistemul cere la momentul potrivit montarea volumului corespunzător. La unele sisteme DEC activitatea de salvare se realizează automat, fără intervenția operatorului sau a administratorului, în momente în care încărcarea sistemului este mai mică (de exemplu pe timpul nopții). Sunt utilizate memorii de masă special construite.

basic input/output system (BIOS) = sistem de intrare/ieşire de bază (BIOS)

Program care execută operații de intrare/ieşire de bază. El realizează teste ale componentelor sistemului cu ocazia lansării acestuia ("boot"), încarcă sistemul de operare de pe disc, conține programe capabile să acceseze echipamentele periferice, etc. În mod normal BIOS-ul este rezident într-o memorie permanentă. El are un rol important în portabilitatea sistemului de operare. Pentru arhitecturi diverse, bazate pe aceeaşi familie de microprocesoare, ceea ce diferă de la un sistem la altul este modul în care se realizează operațiile de intrare/ieşire. Dacă se separă aceste operații în BIOS, aceasta este singura parte a sistemului care trebuie modificată la portarea lui. Procentual aceasta reprezintă o mică parte din totalul programelor de sistem.

batch processing = prelucrare pe loturi

Tehnică de organizare a exploatării sistemului de calcul care se bazează pe acumularea de lucrări (în loturi) care sunt executate împreună pentru creşterea eficienței. În felul acesta, la un moment dat se poate planifica pentru execuție acea lucrare care va utiliza optim resursele disponibile. De exemplu, în cazul existenței unei partiții libere de 256Kb nu se va alege o lucrare care solicită 1Mb sau una care necesită numai 20Kb, iar în cazul unui grad scăzut de utilizare a unității centrale se va planifica o lucrare din clasa celor "CPU intensive".

batch system = sistem de prelucrare pe loturi

Sistem de operare la care prelucrarea se face pe loturi. Sistemul trebuie să dispună de comenzi care se execută neinteractiv. În principiu se rulează în acest fel programe de aplicație lungi sau care utilizează intensiv unitatea centrală sau echipamentele periferice.

benchmark = program de evaluare a performanțelor

Program destinat evaluării performanțelor unor dispozitive hardware sau unor aplicații software. De exemplu benchmark-urile Drystone şi Whetstone sunt destinate măsurării performanțelor unității de calcul în virgulă mobilă. Aceste programe constau în generarea şi executarea unor secvențe de comenzi, măsurarea timpului necesar execuției acestor comenzi şi producerea unor statistici asociate.

best fit = cea mai bună potrivire

Algoritm utilizat în alocarea de spațiu în memoria internă sau auxiliară în cazul unor tehnici de partiționare dinamică. Dintre zonele libere suficient de mari pentru a satisface cererea se alocă aceea care produce cel mai mic spațiu neutilizat (cea mai mică dintre ele). De exemplu pentru alocarea a 100Kb, în cazul unor zone libere de 256kb, 64kb, 128kb, 512kb se va alege zona de 128kb care produce după alocare o nouă zonă liberă de 28kb.

bits per inch (bpi) = biți pe inch

Indicator de măsurare a densității de memorare de informații pe discuri şi benzi magnetice numeric egal cu numărul de biți memorați pe o lungime de un inch (25.4 mm) a pistei. Serveşte la clasificarea echipamentelor (de exemplu: simplă densitate, dublă densitate, densitate mare, etc.). Din dorința de a creşte capacitatea de stocare de informații, densitatea creşte mereu; acest lucru este posibil datorită progreselor tehnologiei.

bits per second (bps) = biți pe secundă

Indicator de măsurare a vitezei de transmisie. El este numeric egal cu numărul de biți transmişi/ recepționați într-o secundă de către un anumit dispozitiv hardware. De exemplu, una din cerințele minimale pentru ca un dispozitiv de afişare să poată fi considerat stație de lucru (workstation) este aceea de a permite desfăşurarea de comunicații cu viteza de 1 Mbps. Pentru utilizarea sistemului de operare de la terminal trebuie setată corect viteza de transfer a acestuia, în concordanță cu caracteristicile sale tehnice (bpi este şi unitate de măsură a vitezei de comunicație pe o interfață serială).

block

Grup de octeți sau cuvinte tratate ca o singură entitate. Unele dispozitive de memorie auxiliară au structură de bloc, la o citire sau scriere transferând un bloc în întregime (dacă sunt necesari numai 2 octeți, se transferă întregul bloc în memorie şi se copiază din zona tampon numai informația utilă). Acestea sunt *blocuri fizice*, spre deosebire de cele *logice*, pe care le organizează utilizatorul pentru că fişierele sunt colecții de astfel de informații structurate.

blocked process = proces blocat

Stare în care se află un proces care nu poate fi executat de către un procesor deoarece procesul aşteaptă producerea unui eveniment. De exemplu, un proces care a lansat o operație de intrare/ieşire poate fi blocat (pentru a nu ocupa inutil procesorul pe perioada execuției acestei operații) şi va aştepta terminarea respectivei operații. În momentul terminării operației, procesul va deveni *pregătit pentru execuție*; ulterior el va putea fi ales de către planificator pentru a primi resursa procesor şi a putea astfel să îşi execute în continuare instrucțiunile. Un proces poate fi de asemenea blocat în aşteptarea eliberării unei resurse temporar indisponibile.

blocking factor = factor de blocare

Numărul de înregistrări logice dintr-un bloc fizic. Pentru a nu risipi spațiul pe suportul extern de informații, mai multe înregistrări logice se grupează într-o înregistrare fizică.

boot (to) = a porni (sistemul de operare)

La pornirea sistemului de calcul, la schimbarea sistemului de operare care îl controlează, sau la reluarea după o cădere, sistemul de operare se încarcă de pe suportul extern şi primeşte controlul după o procedură care poate cuprinde următorii paşi (dar pot exista variații semnificative de la un sistem la altul):

- prin hardware se citeşte automat un bloc de pe disc sau bandă, bloc care cuprinde un mic program de încărcare căruia i se dă controlul;
- programul de încărcare aduce de pe suport extern în memorie acele blocuri care conțin imaginea memorie a nucleului sistemului;
- nucleul primeşte controlul şi execută un număr de proceduri de inițializare, care aduc structurile de date interne în starea corespunzătoare;
- sistemul de operare are controlul integral asupra tuturor resurselor şi activează interfața cu utilizatorul.

În unele cazuri încărcătorul inițial încarcă şi dă controlul unui încărcător mai puternic, în alte cazuri încărcarea are loc de la distanță, prin rețea, etc.

Bos (Business Operating System) = Bos

Sistem de operare proiectat cu intenția de a realiza portabilitatea între diferite maşini a programelor scrise în limbajul COBOL.

BSD - Berkeley Software Distribution = BSD

Sistem UNIX dezvoltat la "University of California at Berkeley", pe baza versiunii 6 a sistemului UNIX produse la laboratoarele Bell. Este una din variantele cele mai răspândite de UNIX (alături de System V - AT&T). Decizia de a oferi gratuit sursele sistemului UNIX unor universități a avut un efect benefic asupra dezvoltării şi impunerii sale atât comercial cât şi în mediul academic. BSD UNIX a cunoscut o dezvoltare continuă şi a produs numeroase îmbunătățiri față de varianta de la care a pornit. Principalele contribuții se referă la realizarea gestiunii memoriei printr-un mecanism de paginare, schimbarea modului de implementare a sistemului de fişiere, introducerea de facilități de comunicație între sisteme UNIX

(sockets) şi a protocolului TCP/IP care a şi devenit un standard de facto. Au fost adăugate şi numeroase utilitare ca: editorul de texte vi, interpretorul de comenzi Cshell, diferite compilatoare. În prezent variantele 4.3 BSD şi 4.4 BSD sunt suportate pe sisteme comercializate de SUN, DEC şi alte firme de prestigiu.

buffer = tampon

Zonă de memorie utilizată de obicei pentru a compensa diferenţa de viteză dintre unitatea centrală de prelucrare şi diferitele dispozitive de intrare/ieşire. Sistemul UNIX are o politică complicată de alocare dinamică de zone tampon utilizate cu rol de memorie "cache". O oprire accidentală a sistemului poate conduce la pierderea unor informaţii stocate în aceste zone tampon, dacă ele nu au fost anterior copiate pe disc.

Litera C

C = limbajul de programare C

Limbaj de programare de nivel înalt, proiectat la începutul anilor '70 de Denis Ritchie la laboratoarele Bell. A fost folosit în rescrierea sistemului de operare UNIX (acesta fiind de fapt scopul principal pentru care a fost proiectat). Este un limbaj utilizat atât de programatorii de aplicaţie cât şi de programatorii sistem. El combină eleganţa şi puterea limbajelor de nivel înalt cu facilităţile limbajelor de asamblare. Limbajele care au stat la baza specificării limbajului C sunt: BCPL (Martin Richards - Cambridge University) şi B (Ken Thompson Bell Labs). Limbajul C este obiectul unui standard ANSI.

C++ = limbajul de programare C++

Este o dezvoltare a limbajului C, realizată de Bjarne Stroustrup în 1980. El furnizează o serie de completări ale limbajului C şi pune la dispoziţia programatorului mecanisme de programare orientată spre obiecte.

Dintre completările aduse de C++ pot fi menţionate:

- operatori de transfer a informaţiei între program şi fişierele de intrare standard şi de ieşire standard;
- transferul prin referinţă al parametrilor de funcţii,
- posibilitatea supraîncărcării semnificaţiei operatorilor,
- noi operatori pentru gestiunea dinamică a memoriei (new, delete).

cache memory = memorie intermediară

Memorie cu performanţe deosebite (dar cu preţ ridicat şi deci cu capacitate mică) interpusă între procesor şi memoria internă. Informaţiile cele mai recent accesate de procesor sunt păstrate aici. Dacă programele îşi concentrează adresările (proprietate numită "locality = localizare a adresărilor") următorul acces la aceeaşi informaţie se va realiza mai rapid, direct în memoria intermediară. Se spune că s-a produs un *ciclu hit*. În caz contrar, informaţia se va aduce din memoria mai lentă (*ciclu miss*). În ultimul timp conceptul a fost extins şi la alte niveluri ale ierarhiei de memorii: de exemplu se face "cash" la disc păstrând informaţiile în zone tampon din memoria internă, sau pe reţea, păstrând fişierul adus de la distanţă (sau o parte a lui) pe discul local, etc.

cancel (to) = a anula (a abandona) acţiunea în curs

Cererile de servicii adresate modulelor sistemului de operare sau de comunicaţii pot fi anulate fie când acestea se află în aşteptare pentru a fi servite, fie când sunt în curs de desfăşurare. Un exemplu tipic este distrugerea unui proces scăpat de sub control sau cu comportament incorect. Cererile lui de intrare/ieşire trebuiesc anulate pentru că ele nu se mai pot desfăşura după dispariţia procesului (şi nici nu trebuie să mai fie servite).

capability = capabilitate

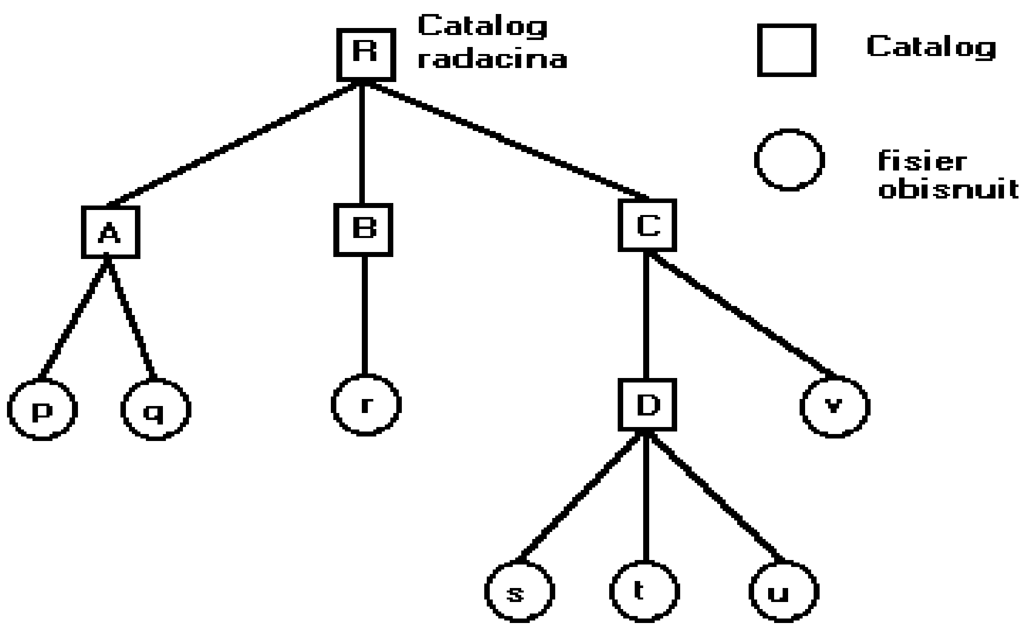
Totalitatea drepturilor de acces pe care le are un proces/utilizator asupra unei colecţii de obiecte protejate din sistem.

capability list = listă de capabilităţi

Lista modurilor în care un proces/utilizator poate accesa obiectele protejate din sistem (sau un subdomeniu al lui). Pentru fiecare obiect se listează acțiunile permise pentru procesul/utilizatorul în cauză. Este o reprezentare a unei linii din matricea drepturilor de acces.

catalogue = catalog (A se vedea şi "directory")

Mecanism folosit în sistemul de gestiune a fişierelor pentru a structura mulțimea fişierelor din sistem şi pentru a adresa simbolic fişierele (prin asocierea de nume complete). De obicei un catalog este tot un fişier ce conține componente numite "intrări în catalog" asociate unor fişiere sau/şi cataloage. Fiecare intrare ar putea conține numele fişierului şi identificatorul său unic în sistem (dar în unele sisteme mai sunt memorate şi alte informații). Cataloagele alcătuiesc o structură ierarhică (arborescentă).



central processing unit (CPU) = unitate centrală de prelucrare

Subsistem al unui sistem de calcul capabil să execute (decodifice, interpreteze) un set de instrucțiuni, să genereze adresele acestora şi să citească/scrie într-o memorie care păstrează programul pe durata execuției acestuia. Procesorul este compus dintr-o unitate de comandă şi una de prelucrare (care asigură executarea instrucțiunilor sub controlul semnalelor de comandă primite de la unitatea de comandă).

Elementele ce caracterizează unitatea centrală de prelucrare sunt repertoriul de instrucțiuni (codul maşină), formatul instrucțiunilor, modurile de adresare, viteza de prelucrare. Unii autori consideră că alături de procesor din unitatea centrală face parte şi memoria internă a sistemului.

Timpul de unitate centrală este o resursă importantă pe care o administrează sistemul de operare prin activitățile de planificare pentru execuție.

characters per inch (cpi) = caractere pe inch

Indicator de măsurare a dimensiunii caracterelor unui dispozitiv de afişare (imprimantă) ce exprimă numărul de caractere afişabile pe orizontală pe un inch. Valori tipice: 10 cpi, 12 cpi, 17 cpi. Diferitele programe care solicită servicii de imprimare, trebuie să specifice în apelurile sistem şi aceste valori de care va fi responsabil driver-ul.

characters per second (cps) = caractere pe secundă (cps)

Indicator de performanță a unui dispozitiv de imprimare numeric egal cu numărul de caractere afişate de către acesta într-o secundă. Valori tipice pot fi de ordinul zecilor, sutelor sau chiar miilor de cps. Acelaşi indicator poate fi folosit şi pentru specificarea vitezei de transfer a informației între două dispozitive hardware.

checkpoint = punct de reluare (într-un program)

Punct de reluare a execuției unui program întrerupt.

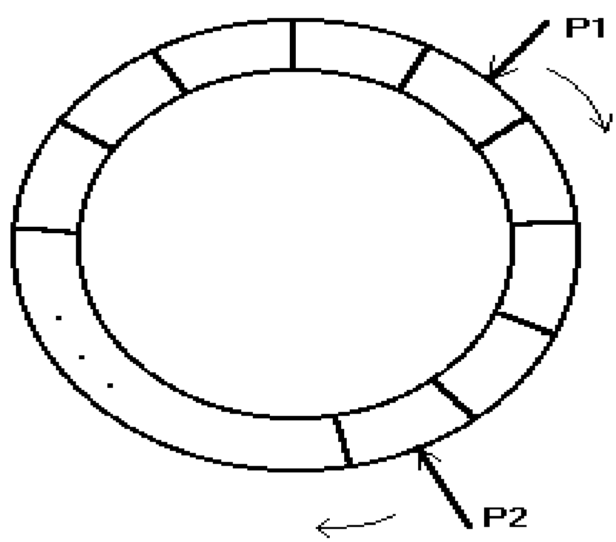
child process = proces fiu

Proces creat dinamic de un alt proces numit *proces tată* (sau *proces părinte*). În sistemul de operare UNIX prin **fork()** se realizează duplicarea *procesului tată*, astfel încât procesul tată şi cel fiu au spații de adresă diferite, dar imediat după creare conținutul celor două spații este identic. Se produce un fenomen de "moştenire" a contextului de execuție şi a fişierelor deschise. În principiu, ulterior, procesul fiu îşi poate schimba conținutul spațiului său de adrese printr-un apel sistem **exec().** Prin aceste mecanisme fork-exec se permite crearea unei ierarhii de procese.

circular buffer = zonă tampon circulară

Zonă de memorie tampon implementată printr-o coadă având asociați doi indicatori. Indicatorul P1 referă elementul unde va fi scrisă următoarea valoare iar P2 referă elementul care va fi citit la următoarea operație de citire din coadă. Se implementează astfel o relație producător/consumator cu zonă tampon limitată. Dacă se consideră că bufferul circular are N elemente, după fiecare operație de citire/scriere a informației din buffer, indicatorul corespunzător va avansa circular, conform relației:

```
p = (p+1) mod N.
```



CLI Command-line interface = interfață orientată linie de comandă

Interfață utilizator care afişează o invitație la dialog şi aşteaptă din partea utilizatorului o comandă sub forma unui text (şir de caractere) care descrie prelucrările pe care trebuie să le facă sistemul.

client = client

Un program utilizat pentru a contacta un alt program numit "server" în vederea obţinerii unui serviciu de la acesta. Serviciul constă dintr-o prelucrare de informaţii sau din regăsirea unor informaţii. Cele două programe se pot afla pe acelaşi sistem de calcul sau pe sisteme diferite, care comunică între ele.

Client/Server Architecture = arhitectură client/server

Mod de proiectare şi implementare a aplicaţiilor (dar şi a unor programe de bază). Un număr de programe "server" (la limită unul singur) sunt specializate în oferirea eficientă a unor servicii de prelucrare de date către un număr de programe "client" cu care comunică în mod disciplinat, potrivit unor protocoale. Programele se pot afla pe sisteme diferite sau pe acelaşi sistem.

close a file (to) = a închide un fişier

Operaţie echivalentă cu eliberarea resursei informaţie de către procesul care o deţine la un moment dat. Se închide canalul de legătură între memorie şi suportul extern, se eliberează structurile de date alocate la deschidere şi eventual se forţează o scriere pe disc a zonei tampon asociate, dacă acest lucru este necesar.

cluster = grup de blocuri de alocare

Grup de blocuri fizice sau logice care se alocă şi se gestionează ca o singură unitate. Unele scheme de alocare a spaţiului ocupat pe disc (dispozitive de memorie externă cu acces aleator), folosesc ca unitate de alocare grupul de blocuri deoarece discurile tind să devină din ce în ce mai mari, adresele de blocuri să ocupe din ce în ce mai mult spaţiu, depăşind uneori posibilităţile de adresare ale procesorului. Alocând unităţi mai mari se gestionează adrese mai puţine şi mai scurte. Un bloc poate coincide cu un sector de pe disc sau poate ocupa mai multe sectoare. Un "cluster" conţine mai multe blocuri şi va conduce la fragmentare internă (pentru 4 octeţi se alocă o unitate de alocare). În cazul sistemului MS-DOS, alocarea spaţiului pe disc este gestionată prin intermediul unei tabele FAT (File Acces Table) al cărui conţinut este stocat în primele sectoare ale pistei 0 a discului. Fiecare intrare în tabela FAT corespunde unui grup de blocuri de alocare. Intrările din tabela FAT sunt folosite pentru a forma liste de "clustere" asociate fişierelor de pe disc. O listă de clustere (L) corespunde unui fişier (F) şi cuprinde numerele tuturor clusterelor ce conţin informaţii din (F). Începutul listei (L) este reţinut în intrarea de catalog asociată lui (F) iar ultimul cluster ce corespunde lui (F) va fi marcat cu o valoare specială.

command file = fişier de comenzi

Fişier ce conţine o succesiune de comenzi ce pot fi tratate de către interpretorul de comenzi ca şi cum acestea ar fi specificate de un utilizator de la terminal. Uneori fişierele de comenzi mai sunt numite şi proceduri (script-uri) shell, fişiere indirecte de comenzi, sau fişiere "batch". Majoritatea limbajelor de comandă moderne sunt şi limbaje de programare, astfel încât în fişier nu se găsesc numai secvenţe de comenzi, ci programe care conţin şi alte structuri de control.

command language = limbaj de comandă

Limbaj folosit de utilizatorul unui sistem de calcul pentru descrierea cerinţelor sale de prelucrare. Este un element caracteristic atât sistemelor de operare interactive, în timp partajat cât şi şi sistemelor de prelucrare pe loturi de lucrări. Majoritatea limbajelor de comandă moderne sunt şi limbaje de programare. Ele pun la dispoziţia utilizatorului comenzi ce pot specifica ordinea de execuţie a altor comenzi, modul de sincronizare şi comunicare între comenzi, proceduri locale etc. Acest limbaj este interpretat, chiar dacă comenzile sunt memorate într-un fişier de comenzi. Interpretorul de comenzi UNIX se numeşte *shell*.

command line = linie de comandă

Şir de caractere, introdus de utilizator şi interpretat de interfaţa de comandă a unui sistem de operare, prin care utilizatorul solicită îndeplinirea unor acţiuni în sistem. Face posibilă utilizarea sistemului de la cele mai simple terminale alfanumerice. În general o linie de comandă este alcătuită din mai multe cuvinte (separate prin spaţii sau caractere TAB). Primul reprezintă numele comenzii, iar următoarele aşa numiţii parametri ai comenzii. Interpretorul de comenzi poate accepta comenzi interne (ale căror funcţii sunt codificate chiar în interiorul sistemului) şi comenzi externe. Acestea din urmă sunt practic programe executabile stocate pe un mediu de informaţie externă. Prin linia de comandă se specifică de fapt încărcarea şi execuţia unui program (sau a mai multor programe) şi eventual modul de efectuare a comunicării şi sincronizării între ele. Pe lângă aceste comenzi imperative pot exista şi comenzi declarative care au ca efect setarea unor câmpuri din structuri de date ale sistemului: determinarea modului de comportament al unui terminal, schimbarea identităţii utilizatorului, etc.

command prompt = simbol de invitație

Şir de caractere afişat de către interfața de comandă a unui sistem de operare ce indică posibilitatea introducerii unei comenzi de către operator. Pot fi asociate şi alte informații cu acest simbol: numele maşinii pe care este deschisă sesiunea la distanță, numele catalogului curent, tipul utilizatorului şi drepturile sale (de exemplu în Bourne-shell UNIX simbolul '$' indică un utilizator obişnuit, iar '#' un utilizator privilegiat numit **root**).

Communicating Sequential Processes (CSP) = procese secvențiale comunicante

Notație propusă de Hoare (1978) pentru proiectarea sistemelor paralele. Conceptele de proces şi comandă gardată au fost încorporate într-un limbaj paralel experimental. Sunt oferite mecanisme de sincronizare bazate pe transferul de mesaje. Sincronizarea a două procese se numeşte *rendezvous* şi conceptul se regăseşte dezvoltat în Ada.

compaction = compactare

Tehnică utilizată în subsistemul de administrare a memoriei într-un sistem de operare. Procedeul constă în fuzionarea tuturor porțiunilor de spațiu liber, prin mutarea (relocarea) zonelor alocate proceselor către adrese mici. Aceeaşi tehnică se utilizează şi pentru disc, pentru a elimina fragmentarea fişierelor şi a spațiilor libere. Se obține un spor însemnat de viteză, micşorând timpii de deplasare şi informațiile memorate pentru regăsirea informațiilor.

compliant = conform

Conformitatea cu anumite standarde este o cerință pe care trebuie să o respecte pachetele de programe de bază şi de aplicații, ca şi echipamentele, pentru a se asigura portarea uşoară a aplicațiilor sau a programelor utilitare. Este o cerință economică majoră, care explică atât importantele eforturi are se fac pentru standardizare, cât şi pe cele de verificare şi atestare a conformității cu aceste standarde.

computer operator = operator la calculator

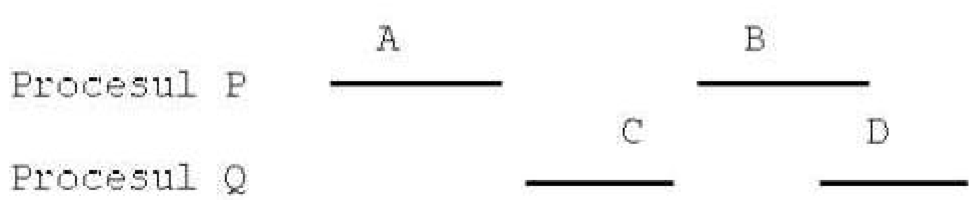
Persoană însărcinată cu supravegherea şi dirijarea funcționării unui sistem de calcul. Este persoana care trebuie să fie familiarizată cu procedurile manuale de utilizare a sistemului: pornirea/oprirea echipamentelor, montarea/demontarea de volume, alimentarea echipamentelor cu hârtie, toner, etc. Operatorul trebuie să stăpânească şi limbajul de comenzi pentru a asigura înlănțuirea activităților în sistem, oprirea lor sau terminarea forțată.

concurrency = concurență

Termen generic pentru suprapunerea în timp a unor activități. Activitățile pot fi independente sau pot implica accesul la resurse comune.

concurrent processes = procese concurente

Două procese P şi Q se numesc *procese concurente* dacă prima operație a unuia începe înaintea terminării ultimei operații a celui de-al doilea. Operațiile pot fi suprapuse în timp, întrețesute sau parțial suprapuse, parțial întrețesute.



concurrent C = C concurent

Extensie a limbajului C care suportă concurenţa. Atât pentru sisteme concentrate cât şi pentru medii distribuite, introducerea de limbaje de programare care suportă concurenţa proceselor s-a realizat prin două metode principale: crearea de limbaje noi sau extinderea unor limbaje de programare secvenţială cunoscute, limbaje care s-au impus deja şi au un număr important de susţinători. Pentru limbajul C sunt cunoscute mai multe astfel de extensii, una bazată pe conceptul de rendez-vous, altele pe comunicarea asincronă între procese prin mesaje.

concurrent Pascal = Pascal concurent

Extensie a limbajului Pascal secvenţial cu procese, monitoare şi clase, realizată de către Per Brinch Hansen şi colaboratorii săi. Primul limbaj care a suportat monitoare, accesul la dispozitivele de intrare/ieşire făcându-se prin apeluri de monitoare. În acest limbaj au fost scrise sistemele de operare SOLO şi TRIO. Din motive de protecţie, pe lângă extensiile amintite, limbajului i-au fost însă impuse şi restricţii, cum ar fi renunţarea la recursivitate. Este unul dintre motivele pentru care limbajul nu s-a impus.

concurrent programming = programare concurentă

Mod de programare specific scrierii sistemelor de operare, conducerii proceselor industriale şi altor aplicaţii în care intervin procese concurente care utilizează resurse comune.

configuration = configuraţie

Totalitatea resurselor de care dispune la un moment dat un sistem de calcul şi modul în care acestea sunt organizate. Exemple de resurse sunt: procesoare, echipamente periferice, memorie internă, memorie cache, memorii auxiliare, suport pentru comunicaţii cu alte sisteme de calcul. Mai multe echipamente periferice pot fi cuplate pe acelaşi controler sau pe cuploare diferite. Un echipament se poate conecta pe o interfaţă serială sau pe multiplexor (iar în anumite situaţii pe o interfaţă paralelă), modul de control fiind diferit. Din configuraţie pot face parte diferite ierarhii de memorii. Comunicaţiile cu alte sisteme de calcul se pot baza pe tehnologii diverse, interfeţele de reţea fiind componente importante ale configuraţiilor actuale. În anumite cazuri se poate identifica o *configuraţie minimală* (minimul de resurse necesar pentru a face funcţionarea posibilă) şi o *cofiguraţie maximală* (cantitatea maximă de resurse ce pot fi controlate de sistem). Unele sisteme de operare permit modificarea dinamică a configuraţiei sistemului de calcul, în timp ce altele necesită reconfigurarea nucleului sau activităţi de administrare cu ocazia modificărilor configuraţiei.

configuration management = administrarea configuraţiei unui sistem

Procesul de identificare, definire, înregistrare şi raportare a resurselor componente ale configuraţiei unui sistem de calcul, precum şi tratarea cererilor de modificare a acestora.

console = consolă

Staţia de lucru a operatorului în sistemele de talie mare ("mainframes") în trecut. Oferea privilegii deosebite persoanei care avea acces la tastatura sa. În UNIX şi în alte sisteme de operare moderne, operatorul (*superuser*) îşi realizează privilegiile la orice terminal prin furnizarea parolei corecte, consola fiind terminalul de la care s-a lansat sistemul. În UNIX consolei i se asociază fişierul special

```
/dev/console
```

Din motive de securitate a sistemului se poate limita activitatea de administrare la un terminal plasat într-un loc sigur. În cazul microcalculatoarelor şi al calculatoarelor personale, ecranul şi tastatura primesc numele de *consolă* şi în unele sisteme de operare li se asociază un nume de pseudo-echipament.

consumer = consumator

Proces care consumă resurse temporare produse de un alt proces numit *producător*. De cele mai multe ori resursele consumate sunt date organizate sub formă de mesaje.

context switch = comutarea contextului

Ansamblu de acțiuni (salvare/restaurare de registre, actualizarea conținutului unor structuri de date sistem) desfășurate în momentul schimbării procesului activ curent (cel care deține controlul procesorului). Când procesorul este liber (procesul activ s-a blocat) sau când se face prelevarea forțată a procesorului, planificatorul de procese alege un nou proces pentru a-i aloca procesorul, se salvează starea procesului întrerupt şi se încarcă starea noului proces.

conversational mode = regim conversațional

Regim de utilizare a unui sistem de calcul prin care utilizatorul se află în comunicare directă cu calculatorul prin intermediul unui terminal. Modul de lucru este interactiv. Utilizatorul introduce comenzi de la tastatură şi primeşte răspunsuri din partea sistemului, putând să corecteze erori de sintaxă sau de logică şi să înlănțuie prelucrările într-o ordine arbitrară. Multe sisteme de operare cu regim interactiv sunt sisteme cu divizarea timpului (*time-sharing*), politica de planificare a proceselor pentru execuție fiind bazată pe alocarea unor cuante de timp egale pe rând tuturor proceselor şi prelevare forțată la expirarea cuantei alocate.

cooked mode = modul "cooked"

În sistemul UNIX este modul normal de lucru (în opoziție cu modul *raw*) în care driver-ul de terminal interpretează caracterele speciale ("erase", "kill", etc), iar întreruperile sunt permise. Mai general, la alte sisteme de operare, acest termen se poate referi la orice mod în care sistemul realizează prelucrări intensive ale datelor, înainte ca acestea să fie transmise programului.

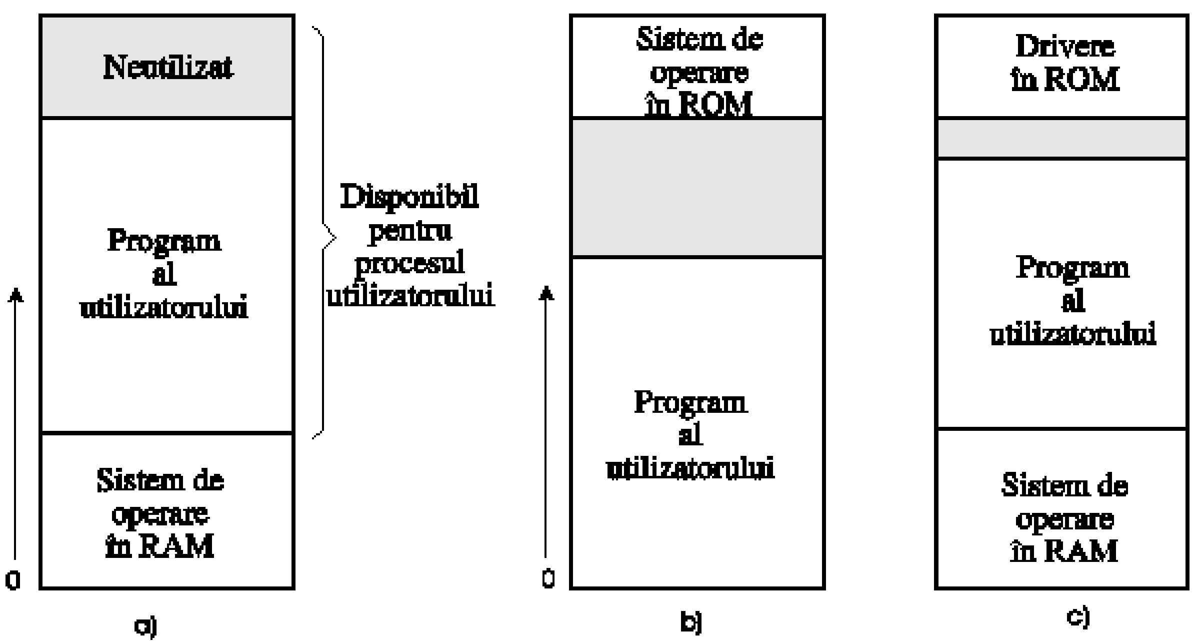
CORAL 66 = CORAL 66

Limbaj de programare pentru timp real. Derivat din limbajele JOVIAL şi ALGOL-60. Limbajul a fost, începând din 1970 şi până la apariția limbajului ADA, standardul adoptat de industria militară britanică.

core = memorie internă

Memorie principală sau memorie RAM. Denumirea datează din timpul când memoriile erau realizate având ca suport inelele de ferită (*ferrite-core*). Termenul este folosit în expresii ca "in core" desemnând informații rezidente în memoria internă.

core map = harta memoriei



Reprezentare intuitivă a conținutului diferitelor zone ale memoriei la un moment dat. Utilizabilă fie în faza de proiectare a modulului de administrare a memoriei, fie pentru monitorizarea stării memoriei cu ajutorul unor programe utilitare. În exemplul de mai jos sunt prezentate hărți ale memoriei pentru diferite modele de administrare a memoriei interne prin alocare

singulară contiguă.

core dump = imagine a conținutului memoriei

O copie a conținutului unei zone de memorie. De exemplu, imaginea memoriei ocupate de un proces, produsă când procesul este terminat forțat ca urmare a unei erori interne sau a unei intervenții operator. A se vedea "dump", "abort".

CP/M (Control Program for Microcomputers) = CP/M

Sistem de operare pentru microcalculatoare bazate pe microprocesoare pe 8 biți Intel 8080 şi Z-80 scris de Gary Kildall (fondatorul companiei Digital Research). A fost înlocuit de MS-DOS după apariția în 1981 a IBM PC.

CPU time = timp de unitate centrală

Durată de timp în care un proces a avut la dispoziție resursa procesor (i.e. procesorul a executat efectiv instrucțiunile respectivului proces). În restul timpului procesul este blocat în aşteptarea terminării unor operații de intrare/ieşire sau eliberării unor resurse, sau aşteaptă să i se aloce procesorul (se află în starea "*pregătit pentru execuție*").

cracker = "cracker"

Utilizator rău intenționat care îşi petrece timpul încercând să depăşească barierele de securitate specifice unui sistem de operare. În mod normal acest tip de utilizator nedorit nu produce pagube, dar uneori poate să şi aducă prejudicii importante sistemului (ştergere de fişiere, modificarea lor sau chiar distrugerea întregului sistem). Formă de vandalism în utilizarea facilităților oferite de un sistem de calcul. De obicei cunoştințele teoretice ale unui "cracker" au un nivel coborât, el imitând (preluând) unele rețete citite în cărți, pe Internet sau văzute la alți utilizatori.

crash = cădere (în pană)

Cădere neaşteptată, bruscă a sistemului. În cazul unui sistem de operare oprirea poate fi voluntară. Unele dintre cauzele care pot provoca o astfel de oprire sunt defectarea discului de pe care se încarcă componentele sistemului, sau detectarea unor inconsistențe în structurile de date interne ale sistemului. Chiar în cazurile în care erorile ar putea fi ignorate, se preferă oprirea sistemului pentru a se limita întinderea unor eventuale distrugeri ale informațiilor. Uneori se salvează o imagine a memoriei ("dump") care permite o analiză ulterioară a circumstanțelor în care s-a produs căderea.

critical region = regiune critică

Notație în limbaje de nivel înalt pentru realizarea excluderii mutuale a accesului proceselor la variabile utilizate în comun. Construcția:

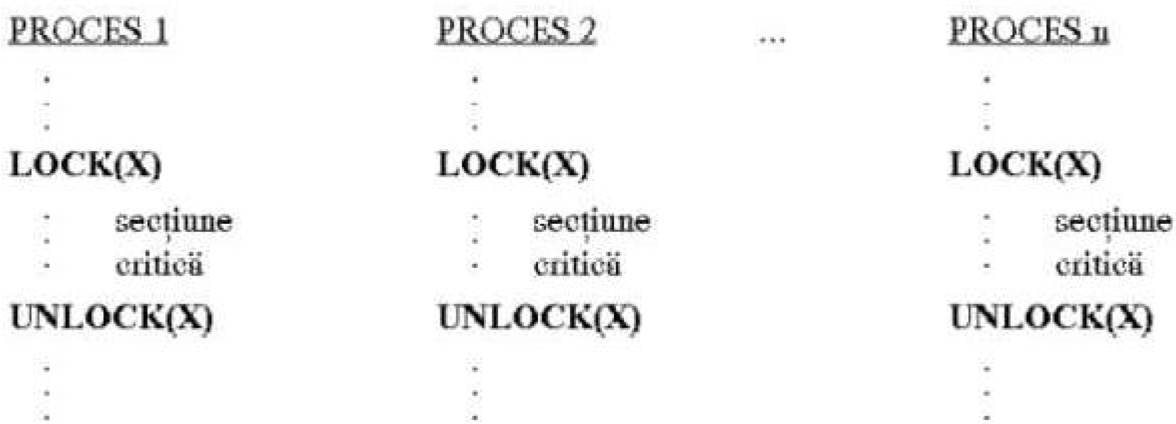
```
region v do S;
```

apare într-un proces P care trebuie să execute operațiile *S* în care se accesează componentele variabilei *v* declarată ca variabilă comună (**shared**). Se impune planificarea pentru execuție a procesului într-o secțiune critică. Alte procese care doresc să facă acces la aceeaşi variabilă vor trebui să aştepte ieşirea procesului P din secțiunea critică.

critical section = secțiune critică

Secvență de cod nereentrant care poate fi executată de către un singur program la un moment dat. În această parte a programului procesul accesează zone de memorie partajată, fişiere utilizate în comun cu alte procese sau alte resurse comune a căror utilizare fără excludere mutuală poate conduce la condiții de cursă. Este necesar ca execuția secțiunii critice să se termine într-un timp cât mai scurt (pentru a permite şi altor procese aflate în aşteptare să intre în secțiune critică). Decizia de a permite accesul unui proces în secțiune critică trebuie luată într-un timp finit (un proces nu poate fi amânat la infinit datorită sosirii unor noi procese cu prioritate mai mare).

Realizarea unei secțiuni critice cu ajutorul mecanismelor de sincronizare de tip octet de blocare se face ca mai jos:



Se poate controla o secțiune critică utilizând un semafor *mutex* cu valoarea inițială 1 ca în exemplul următor:

P(mutex)

sectiune_critică

V(mutex)

Secțiunea critică poate fi văzută în modul cel mai general ca o resursă.

cron = cron

Proces de tip daemon asociat ceasului în sistemul de operare UNIX. El are ca funcție executarea unor comenzi la anumite momente, după cum s-a specificat într-un fişier (/etc/crontab).

csh (C-shell) = csh

Una dintre interfețele de comandă specifice sistemului de operare UNIX. Acest shell este caracterizat de comenzi de control a fluxului execuției având o sintaxă similară cu instrucțiunile limbajului de programare C (*if, foreach, switch*). Interpretorul C-shell a pus pentru prima oară la dispoziția utilizatorilor facilități pentru gestiunea listei "istoric" a comenzilor date în timpul unei sesiuni de lucru, ca şi comenzi pentru manipularea proceselor care rulează în background sau în foreground. C-shell a fost creat de William Joy.

CTSS - Compatible Time-Sharing System = CTSS

Sistem de operare cu divizarea timpului implementat în 1962-1963 la MIT ca un experiment în realizarea de sisteme interactive (maşina de calcul utilizată a fost un IBM 7092 modificat). În cadrul proiectului Multics, CTSS a fost utilizat ca prima încercare cunoscută de a implementa un sistem de operare în limbaj de nivel înalt lucrând sub un mediu cu multiacces.

cu (Call UNIX) = cu (comandă UNIX)

Utilitar UNIX care asigură facilități de terminal virtual pe Ethernet, linie directă sau modem.

curses = curses

Bibliotecă de rutine în sistemul de operare UNIX care oferă posibilitatea navigării cu ajutorul unui cursor pe terminale cu tub catodic. Se gestionează în regim semigrafic afişarea pe terminale alfanumerice şi se pot defini ferestre cărora li se asociază atribute diverse (culoare, video-invers, chenar, etc.). Se pot realiza programe *independente de terminal,* caracteristicile curente ale terminalelor suportate fiind codificate în bazele de date de terminale

```
termcap
```

sau/şi

```
terminfo
```

Ele sunt preluate şi folosite de rutinele bibliotecii curses.

customization = adaptare

Adaptarea unui pachet de programe sau a unei configuraţii hardware la cerinţele specifice utilizatorului. În momentul instalării, sau ulterior, prin activitatea de administrare se modifică anumiţi parametri ai produsului, astfel încât el să răspundă cât mai bine scopului pentru care a fost achiziţionat, tipului de aplicaţii pentru care este utilizat sau unor caracteristici ale echipamentelor.

cylinder = cilindru

Ansamblul pistelor pe care se află capetele unui disc într-o poziţie dată, fixă a braţului. În sistemele de gestiune a fişierelor din trecut, alocarea spaţiului la crearea unui fişier se realiza specificând un număr de cilindri. A se vedea "disk".

Litera D

daemon = demon

Proces care rulează în fundal (*background*) pentru a aştepta cereri şi a îndeplini sarcinile legate de servirea acestora. De obicei daemonii sunt porniţi la iniţializarea sistemului de operare (*boot*) şi sunt opriţi odată cu acesta (*shut-down*), dar pot fi şi excepţii de la această regulă (la dezactivarea unui serviciu se opreşte daemonul corespunzător). În sistemul de operare UNIX există numeroşi daemoni, ca de exemplu

```
cron, rshd, telnetd, ftpd, nfsd
```

, care îndeplinesc diferite servicii cum ar fi execuţia la un anumit moment a unei comenzi, execuţia la distanţă a unei comenzi, conectarea la distanţă, transferul de fişiere. Pentru tratarea de cereri venite de la alte noduri ale reţelei se preferă actualmente ca daemonii să nu ruleze permanent, ci să fie lansaţi de către un daemon unic, numit

```
inetd
```

; şi în alte sisteme de operare s-au utilizat daemoni. Un program care dorea să utilizeze imprimanta activa implicit daemonul de spooling şi astfel putea ignora problemele legate de alocarea perifericului sau de conducerea acestuia. Daemonii erau lansaţi automat de către sistem şi puteau fi regeneraţi la anumite intervale. Aceste programe nu erau apelate explicit. Ele aşteptau în starea "dormant" îndeplinirea unor anumite condiţii. Numele daemon este un acronim de la "Disk And Execution MONitor".

data security = securitatea datelor

Cerinţă pe care trebuie să o îndeplinească orice sistem de prelucrare a datelor. În particular, un sistem de operare gestionează printre alte resurse şi informaţii aparţinând unor utilizatori diferiţi. Una dintre sarcinile sistemului este aceea de a asigura integritatea acestor date, împiedicând accesul neautorizat al unor programe eronate sau rău intenţionate.

DCL (Digital Command Language) = DCL

Limbaj de comandă (şi de comenzi indirecte) specific sistemului de operare VAX/VMS produs de firma DIGITAL.

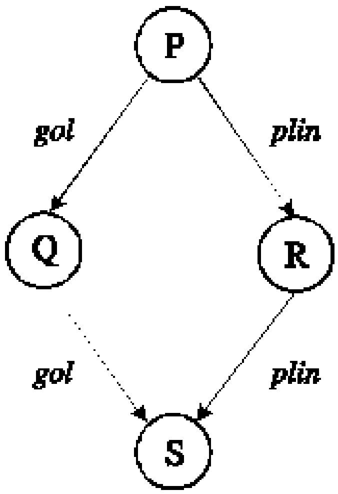
DDE (Dynamic Data Exchange) = DDE

Metodă realizată de Microsoft care permite utilizarea inteligentă de către o aplicaţie a datelor create de către o altă aplicaţie.

Utilizată printre altele în Microsoft Windows. A se vedea şi OLE.

deadlock = interblocare

Situație în care o mulțime de procese nu pot progresa în execuție deoarece fiecare dintre ele aşteaptă un eveniment pe care numai unul dintre procesele din această mulțime îl poate produce. Evenimentul este de cele mai multe ori asociat cu eliberarea unei resurse. Aşteptarea poate fi circulară, implicând mai multe procese.



În exemplul prezentat în figura alăturată, patru procese P, Q, R şi S comunică prin mesaje, fiind conectate prin zone tampon:

P este blocat pentru că aşteaptă ca R să recepționeze (buffer plin);

R este blocat pentru că aşteaptă ca S să recepționeze (buffer plin);

S este blocat pentru că aşteaptă ca Q să emită (buffer gol);

Q este blocat pentru că aşteaptă ca P să emită (buffer gol).

Sistemul se află în starea de blocaj.

S-a demonstrat (Coffman 1971) că pentru a se produce interblocare, într-un sistem trebuiesc îndeplinite patru condiții (dacă una singură dintre aceste condiții nu se realizează blocajul reciproc nu este posibil, iar dacă sunt îndeplinite toate blocajul nu este sigur că va apare):

1. utilizarea resurselor în excludere mutuală
2. planificarea resurselor fără prelevare forțată
3. alocare parțială a resurselor
4. aşteptare circulară pentru resurse.

Sistemele de administrare a resurselor trebuie să rezolve trei probleme legate de blocajul reciproc:

1. prevenirea apariției interblocării
2. detectarea situațiilor de blocaj
3. recuperarea sistemului din starea de blocaj.

deadly embrace = interblocare

Situație de blocaj reciproc identică cu cea descrisă de termenul *deadlock*, dar în care sunt implicate exact două procese. De exemplu, procesele A şi B sunt executate într-un sistem care dispune de o bandă magnetică şi o imprimantă care nu se virtualizează. Ambele procese au nevoie atât de imprimantă cât şi de banda magnetică şi nici una dintre aceste resurse nu se

poate preleva forțat. Procesul A după ce a primit banda magnetică este blocat pentru că imprimanta este alocată proocesului B, care este la rândul său blocat în așteptarea benzii magnetice deținute de A.

degree of multiprogramming = grad de multiprogramare

Numărul n de programe încărcate simultan în partiții ale memoriei în sistemele cu multiprogramare (tehnică utilizată în sisteme mai vechi pentru creşterea gradului de utilizare a unității centrale de prelucrare). Pentru fiecare program se crea un singur proces, aceste procese participând la competiția pentru acapararea procesorului. În sistemele moderne pentru fiecare program se pot crea mai multe procese.

deallocate (to) = a dealoca (elibera)

În contextul sistemelor de operare eliberarea unei resurse deținute de către un proces sau un program al unui utilizator. Un proces poate elibera explicit o resursă de care nu mai are nevoie, unele resurse pot fi eliberate forțat (*pre-emption*), iar la sfârşitul unui program sau la terminarea sa forțată toate resursele deținute sunt eliberate.

dedicated = dedicat

1. Resursă alocată unui proces o perioadă nedefinită de timp.
2. Sistem dedicat: sistem de operare destinat unei aplicații particulare (de exemplu conducerii unui proces industrial, monitorizării stării unui bolnav aflat la terapie intensivă, etc.).

default drive = unitate implicită

Unitatea de disc, de bandă sau altă memorie auxiliară considerată implicit atunci când în specificatorul de fişier nu se indică explicit numele echipamentului pe care se află fişierul. În foarte multe sisteme de operare, adresarea simbolică a fişierelor se face cu nume compuse care conțin informații diverse (nume fişier, extensie, versiune, etc.) la care se adaugă cele de localizare a fişierului (numele maşinii, a discului, calea în ierarhia de cataloage). Unitatea implicită poate fi stabilită prin administrarea sistemului şi poate fi modificată prin comenzi date de utilizator. Sistemele din familia UNIX au specific faptul că se lucrează cu un arbore unic de fişiere care "ascunde" echipamentul pe care se află localizat fişierul; numele acestui echipament nu face parte din specificatorul de fişier.

demand paging = paginare la cerere

Tehnică de administrare a memoriei care asigură memorie virtuală. Spațiul de adrese al programului este împărțit (în mod transparent pentru utilizator) în zone de lungime fixă, (tipic de 1Kb, 2Kb, 4Kb, 8Kb, 16Kb) numite *pagini*. Memoria fizică este la rândul ei împărțită în *blocuri* de lungime fixă, egale cu paginile. Teoretic orice pagină se poate încărca în orice bloc (cu excepția celor rezervate pentru sistemul de operare). O pagină se încarcă numai "la cerere", adică numai în momentul în care se adresează o locație din spațiul ei de adrese. Este posibil ca ea să înlocuiască o altă pagină care a ocupat anterior blocul care i-a fost alocat. Aceasta implică utilizarea unui algoritm de înlocuire de pagini. După încărcare, pagina va rămâne un timp în memorie până ce va fi la rândul ei înlocuită. O paginare la cerere eficientă implică existența unor mecanisme hardware corespunzătoare.

demon = demon

1. Porțiune a unui program care nu este apelată în mod explicit ci rămâne dormantă în aşteptarea îndeplinirii unor anumite condiții (aceasta este semnificația asociată termenului la MIT). Astfel de demoni sunt folosiți în aplicații din domeniul inteligenței artificiale pentru a implementa reguli de inferență. În acest context se consideră că *"demon"* este un proces în cadrul unui program, în timp ce *"daemon"* este un program executat într-un sisteme de operare.
2. În afara MIT, în special în lumea sistemelor de operare UNIX, termenii *"demon"* şi *"daemon"* sunt considerați echivalenți. A se vedea şi daemon.

Desktop Manager = Desktop Manager

Interfață utilizator la servicii sistem orientată pe menu-uri şi iconițe. Permite executarea de aplicații şi utilizarea sistemului de fişiere fără întrebuințarea explicită a limbajului de comandă al sistemului de operare.

device driver = program de comandă (driver)

Program de sistem, componentă a modulului pentru administrarea echipamentelor periferice, ce conține cod dependent de dispozitiv. Conduce un anumit dispozitiv sau un anumit tip de dispozitive. Are rolul de a realiza efectiv conducerea echipamentului periferic prin:

- crearea şi lansarea operațiilor de intrare/ieşire
- tratarea întreruperilor
- optimizarea funcționării echipamentului periferic
- anularea operațiilor în aşteptare
- tratarea unor condiții de eroare
- tratarea unor condiții de eroare
- tratarea unor condiții speciale (de exemplu reluarea după o cădere de tensiune)

device independent = independent de dispozitiv

1. Cerință la care trebuie să răspundă programele de intrare/ieşire. Un program trebuie să fie capabil să lucreze cu fişiere de pe disc flexibil sau hard disk fără să necesite vreo modificare.
2. Caracteristică a aplicațiilor dezvoltate sub un anumit sistem de operare prin care se asigură portabilitatea acestora pe o gamă largă de dispozitive de intrare/ieşire (de regulă terminale). Pentru a asigura independența de dispozitiv a aplicațiilor grafice, acestea sunt implementate pe două niveluri:
   - nivelul independent de dispozitiv care oferă aplicației un set de funcții de nivel înalt de gestiune a unui echipament virtual de afişare/interacțiune,
   - nivelul dependent de echipamentele fizice ale sistemului grafic.

În sistemul de operare UNIX aplicațiile dezvoltate utilizând una dintre variantele bibliotecii

`curses`

folosesc rutine care citesc caracteristicile diferitelor terminale stocate în fişierul

`/etc/termcap`

(numărul de linii/coloane, dacă terminalul suportă sau nu mecanismul backspace de ştergere a ultimului caracter afişat, secvențele de control pentru afişare video-invers sau intermitent, etc.). Acest fişier este o bază de date conținând informații despre un mare număr de terminale, de la cele mai simple la cele mai sofisticate, de proveniențe diverse. Plasarea informației dependente de dispozitiv într-un fişier care poate fi modificat cu uşurință de utilizator permite adaptarea aplicațiilor pentru diferite tipuri de terminale şi considerarea unor echipamente noi. Mecanisme asemănătoare sunt

`terminfo`

ca şi driverul ANSI.SYS folosit în diferite sisteme (de exemplu MS-DOS) pentru a scrie aplicații care să utilizeze în regim semigrafic un dispozitiv de afişare.

`/dev/null = /dev/null`

Pseudo-dispozitiv de intrare/ieşire (aceasta este forma din UNIX, dar conceptul există şi în alte sisteme de operare) caracterizat prin faptul că orice operație de intrare/ieşire se termină cu succes dar nu are nici un efect asupra sistemului.

direct access storage device (DASD) = (echipament de) memorie cu acces direct

Dacă se notează cu Tik timpul de acces la blocul *k* de informații dintr-o memorie auxiliară dacă ultimul acces s-a făcut la blocul *i*, echipamentele cu acces direct sunt cele la care Tik poate avea variații mici. La discul magnetic de exemplu, timpul de acces la o înregistrare aflată la o distanță de 3 piste față de poziția curentă a capetelor va fi mai mic decât timpul de acces la o înregistrare situată cu 10 piste mai departe, dar nu va diferi foarte mult de acesta. Dispozitivele cu acces direct sunt în marea lor majoritate dispozitive cu capete mobile, dar s-au construit şi dispozitive cu capete fixe.

directive = directivă

Nume dat apelurilor sistem în anumite sisteme de operare (de exemplu RSX-11M). Prin directive date nucleului, programul utilizator obține servicii din partea sistemului de operare.

directory = catalog

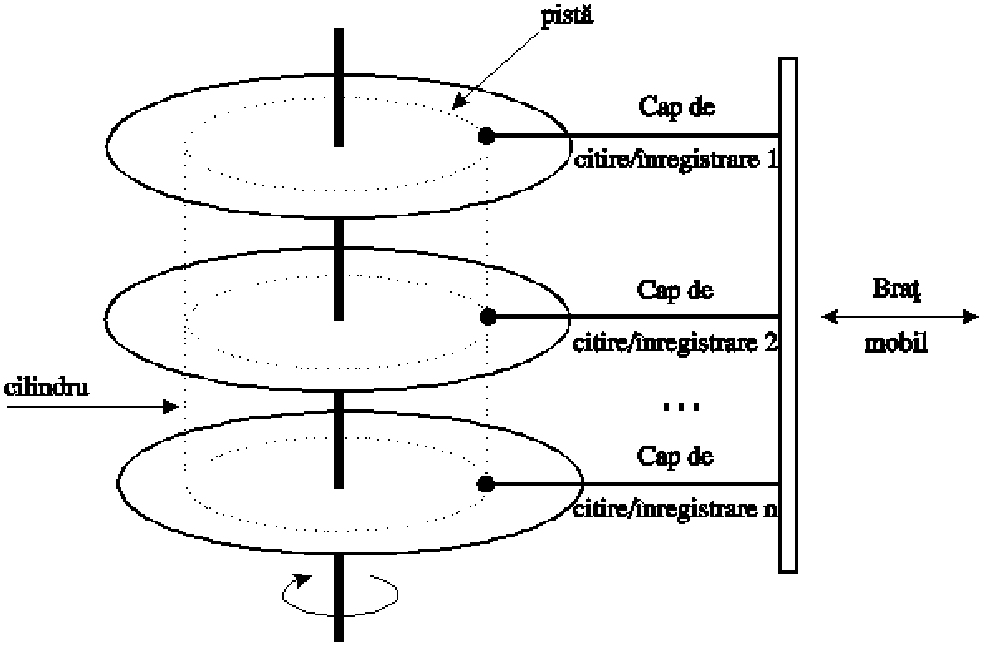
Posibilitate de a grupa fişierele după utilitatea lor, apartenența la un utilizator sau la o aplicație; această posibilitate este oferită de foarte multe sisteme de fişiere. (A se vedea "file directory" şi "catalogue").

(to) disable interrupts = a dezactiva întreruperi

Prima soluție de realizare prin hardware a excluderii mutuale. Tehnică prin care sistemul de operare poate realiza o secțiune critică într-un sistem cu un singur procesor (de preferință la nivelul nucleului). Prin dezactivarea întreruperilor sistemul nu mai poate lua cunoştință de producerea unor evenimente externe (inclusiv sfârşitul unei cuante de timp) şi nu poate pierde controlul în timp ce tratează o secțiune critică.

disk = disc

Dispozitiv de memorie externă care păstrează nucleul sistemului de operare, componentele sistemului şi alte fişiere de date sau conținând programe în diferite limbaje şi formate (sursă, obiect, executabile). Există variante de discuri flexibile sau dure, fixe sau amovibile, cu capete mobile sau fixe (rare în prezent), cu diferite capacități şi densități de stocare a informațiilor. În figura alăturată este prezentat un astfel de dispozitiv format din mai multe discuri plasate pe acelaşi ax care se roteşte cu viteză constantă. Suprafața discului este acoperită cu un material cu proprietăți magnetice care serveşte la memorarea şi redarea informațiilor. În fața fiecărei fețe este poziționat un cap de citire/înregistrare. Suprafața discului este împărțită logic în piste. Când discul se roteşte şi capul rămâne fix, informațiile de pe pistă trec prin fața capului şi pot fi citite sau modificate. Toate capetele sunt legate solidar cu un braț mobil şi se pot muta de pe o pistă pe alta. Fiecare poziție a brațului se numeşte *cilindru*. Pe fiecare pistă informația este memorată prin înregistrare magnetică în blocuri. De regulă aceste blocuri au lungime constantă, specifică fiecărui dispozitiv şi se numesc *sectoare* datorită formei lor de sector circular. Fiecare sector poate fi citit sau scris separat, printr-o operație indivizibilă. Tipic, sectoarele au 256, 512, 1024 octeți. La fiecare operație de citire/scriere a unui sector, dacă acesta are 512 octeți, se transferă exact 512 octeți chiar dacă nu sunt toți necesari; se impune utilizarea unei zone tampon. Între blocuri există separatori a căror lungime depinde de turația discului şi de lungimea pistei, fiind constanți pentru o pistă dată. Din această cauză pistele interioare, mai scurte, sunt mai expuse la erori decât cele exterioare şi nu este indicat să conțină informații vitale ale sistemului de fişiere. Există şi discuri cu blocuri de lungime variabilă, dar tehnica aceasta este mult mai scumpă. Fiecare sector sau bloc este compus dintr-o zonă antet (în care printre altele este memorată adresa sa unică pe disc) şi o zonă de date (în care se memorează datele propriu-zise). Pentru a preciza poziția unui bloc particular, trebuie specificată poziția brațului (numărul de cilindru), capul selectat (numărul de pistă) şi numărul de ordine al sectorului pe pistă (număr de bloc). Prin concatenarea acestor trei numere se obține adresa unică a blocului pe disc.



disk formatting = formatarea discului

1. În majoritatea sistemelor de operare formatarea unui disc este operaţia prin care se rescriu informaţiile din zonele antet ale sectoarelor (blocurilor fizice). Eventualele structuri de sistem de fişiere se pierd cu acestă ocazie. Formatarea urmează să fie succedată de o verificare a blocurilor de date şi de crearea unui sistem de fişiere, operaţii care nu au legătură cu faza de formatare.
2. În contextul sistemului MS-DOS, operaţie prin care se realizează iniţializarea unui disc la un format acceptat de sistemul de operare. De obicei la formatare se analizează discul pentru a detecta unităţile de alocare a spaţiului defecte. Acestea vor fi marcate. Pe disc se creează o structură de fişiere, iar la cerere se poate copia şi un sistem autoîncărcabil ("bootabil"). Această formatare la nivel logic poate fi precedată şi de o formatare de nivel scăzut, (la nivel fizic se realizează şi iniţializarea antetului fiecărui sector).

disk head = cap de citire/scriere (pe disc)

A se vedea "disk".

disk operating system = sistem de operare pe disc

Termen generic pentru un sistem de operare încărcat de pe disc. În momentul lansării sistemului de operare (*boot*) se încarcă în memorie nucleul rezident, iar apoi şi alte componente ale sistemului, pe măsură ce sunt necesare. În afara sistemelor de operare pe disc, există sisteme a căror încărcare se poate realiza de pe bandă magnetică, de la distanţă pe linie de comunicaţie, etc. Există şi sisteme dedicate care se găsesc în memorii permanente şi sunt lansate la punerea sub tensiune a sistemului.

diskless = fără discuri

Sistem de calcul fără disc, de regulă parte componentă a unei reţele de calculatoare (sau a unei arhitecturi de tip cluster). În calitate de client îşi poate stoca şi obţine informaţiile de pe discuri gestionate de către alte sisteme componente ale reţelei.

dispatcher = planificator (de acţiuni)

În anumite sisteme de operare numele dat modulului care face planificări sau care primeşte apelurile sistem şi activează componentele însărcinate cu tratarea lor.

DP (Distributed Processes) = DP

Primul limbaj de programare bazat pe apeluri de procedură la distanță (RPC). Acest concept de programare concurentă a fost introdus de Per Brinch Hansen (1978). A avut o influență importantă asupra mecanismelor de comunicație din Ada.

distributed operating system = sistem de operare distribuit

Sistem care rulează pe o colecție de mașini care, deşi nu au o memorie comună, dau utilizatorilor impresia existenței unui singur calculator (conform Tanenbaum). Utilizatorii ignoră pe ce procesor se execută procesele lor sau pe ce disc se află fişierele lor; aceasta cade în întregime în seama sistemului. Unii autori definesc sistemul de operare distribuit prin existența unei singure imagini a sistemului, diferitele sale componente fiind la un moment dat distribuite pe diferite procesoare. Alți autori consideră că un sistem distribuit este un sistem care se execută pe o colecție de mașini interconectate în rețea, care acționează ca un singur procesor virtual.

DLL (Dynamic Link Library) = DLL

O mulțime de funcții şi proceduri care oferă o metodă de acces dintr-un program la colecții de date create de alte programe de aplicație. Astfel de mecanisme sunt utilizate de exemplu de aplicații Windows Microsoft. Se oferă un mijloc de comunicare între programe dezvoltate în medii de programare diferite.

DOS (Disk Operating System) = DOS (Disk Operating System)

Sistem de operare dezvoltat de către Microsoft şi utilizat pe calculatoare personale IBM şi compatibile. Este automat încărcat de pe disc în memorie la pornirea calculatorului, prin utilizarea unei proceduri de 'bootstrap' din BIOS (Basic Input Output System) aflat într-o memorie permanentă ROM. Poate fi reîncărcat şi la cerere (de exemplu prin combinația de taste Ctrl-Alt-Del). Este un sistem simplu, monoutilizator. A se vedea MS-DOS.

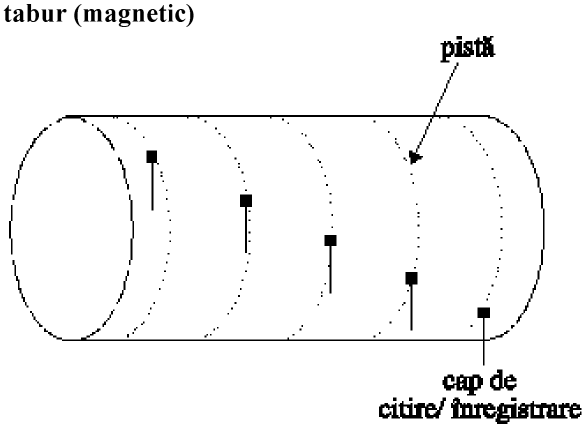
dot file = fişier al cărui nume începe cu "."

Prin convenție în sistemul de operare UNIX un fişier al cărui nume începe cu "." este considerat invizibil (la listarea normală a conținutului unui catalog). De obicei astfel de fişiere conțin informații de configurare sau inițializare necesare diferitelor programe de aplicație. Prin editarea acestor fişiere utilizatorul poate modifica comportamentul programelor cărora le sunt asociate. De exemplu fişierul

```
.profile
```

conține comenzi şi setări care se execută la deschiderea unei sesiuni de către un utilizator care are asociat ca interpretor de comenzi Bourne-shell.

drum = tabur (magnetic)



Dispozitiv de memorie externă cu capete fixe (amintit numai pentru caracterul său istoric). Avea informațiile memorate pe piste circulare plasate pe suprafața exterioară a unui cilindru. Cu fiecare pistă era asociat un cap fix; adresarea unui bloc se realiza indicând numărul capului (pistei) şi al blocului (sectorului pe pistă). Tamburul magnetic a fost utilizat drept suport pentru sisteme de memorie virtuală. Pagini din programe sau programe întregi erau evacuate cu mare viteză pe tambur şi reîncărcate ulterior de acolo. În primul caz, un bloc de pe tambur corespundea ca mărime cu o pagină, iar echipamentul se

numea *tambur de paginare*.

dumb terminal = terminal neinteligent

Terminal cu structură deosebit de simplă, fără capacitate de prelucrare locală a informației recepționate sau transmise. Noțiunea de "inteligență a terminalelor" are un catacter relativ. Un terminal considerat "inteligent" pe o anumită treaptă de dezvoltare a tehnologiei poate fi repede depăşit şi devine "neinteligent".

dump (to) = a copia (o zonă de memorie)

Copie a unei zone de memorie internă sau auxiliară relizată pe un echipament periferic; de cele mai multe ori are scopul de a permite analiza conținutului acestei zone după terminarea cu eroare a programului. Existența depanatoarelor simbolice moderne, ca şi posibilitățile proprii de depanare ale limbajelor de nivel înalt face să scadă importanța acestei tehnici de depanare. A se vedea şi "crash". Uneori copia se face în intenția de salvare a unor informații. A se vedea "backup" şi "incremental dump".

dynamic linking = legare dinamică

Legarea dinamică sau legarea "întârziată" reprezintă o editare de legături efectuată la momentul execuției unui program. Execuția programului începe fără parcurgerea fazei de editare de legături. Modulele invocate sunt localizate, legate şi încărcate numai în momentul utilizării lor. În felul acesta se poate realiza punerea la punct a programului pe porțiuni, chiar înaintea scrierii sale complete, ceea ce conduce la creşterea productivității în implementarea de aplicații complexe şi facilitează lucrul în echipă. Tehnica aceasta se poate aplica în cazul administrării memoriei cu segmentare. Programul este alcătuit din segmente care se referă între ele prin nume simbolice. Dacă în timpul execuției programului se ajunge la o astfel de referință nesatisfăcută, segmentul va fi încărcat în acel moment (dacă a fost scris şi compilat) iar numelui simbolic i se va asocia o valoare numerică.

Litera E

emacs = emacs

Editor de texte orientat ecran, utilizat sub sistemele de operare UNIX, VMS, etc. A fost elaborat de Richard Stallman în cadrul proiectului GNU şi este distribuit de Free Software Foundation. Acest editor este extensibil (i se adaugă cu uşurință noi funcții), uşor reconfigurabil de către utilizator, autodocumentat; emacs este un editor "what you see is what you get" cu afişare în timp real. Se permite lansarea de subprocese (pentru compilare sau poştă electronică) oferind astfel un mediu de dezvoltare de programe. Este un editor de texte programabil care conține un subsistem LISP. Denumirea emacs este un acronim de la **E**diting **MAC**ro**S**.

EMM (Expanded Memory Manager) = EMM

Subsistem ce realizează gestiunea şi comutarea blocurilor de memorie expandată. A se vedea "expanded memory".

EMS (Expanded Memory Specification) = EMS

Numele dat standardului pentru schema de memorie expandată dezvoltată de companiile Lotus, Intel şi Microsoft pentru microprocesorul Intel 8088, standard cunoscut şi sub numele LIM EMS (Lotus/Intel/Microsoft Expanded Memory System). A se vedea şi "expanded memory".

emulation = emulare

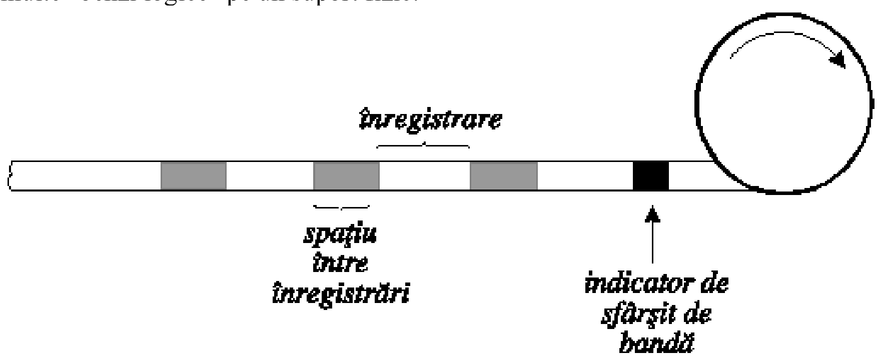
Realizarea funcțiilor unui echipament cu ajutorul altui echipament. Un calculator personal sau o stație de lucru poate să emuleze un terminal, asigurând conectarea la un nod al unei rețele. Un calculator, prin schimbarea conținutului memoriei de microprogram, poate să emuleze instrucțiunile unui alt calculator, etc.

end of file (EOF) = sfârşit de fişier

1. Înregistrare specială care marchează sfârşitul unui fişier.
2. Sistemele de fişiere oferă posibilitatea ca prin apeluri sistem să se semnaleze detectarea sfârşitului unui fişier aflat în exploatare. Valoarea întoarsă de funcţiile C de intrare din fişier când s-a ajuns la sfârşitul fişierului poartă numele simbolic *eof*.
3. În UNIX, caracterul mapat de driverul de terminal pe condiţia "sfârşit de fişier" (Ctrl-D, EOT, etc).

end of tape = sfârşit de bandă

Marcaj aplicat pe suprafaţa unei benzi magnetice pentru a putea fi detectat de unitatea de bandă magnetică. În acest fel se indică sfârşitul zonei de pe bandă pe care pot fi stocate informaţii. Prin aplicarea mai multor astfel de marcaje se pot realiza mai multe "benzi logice" pe un suport fizic.



environment = mediu

Contextul în care se execută un program şi în care acesta primeşte asistenţă din partea sistemului de operare. Instalarea unui pachet de programe poate să solicite definirea sau modificarea unor variabile de mediu. Printre variabilele de mediu se pot număra: cele care definesc căile în ierarhia de fişiere pe care se încearcă localizarea unor fişiere, tipul terminalului de la care se lucrează şi caracteristicile sale, editorul de texte care se utilizează pentru pregătirea mesajelor de poştă electronică, promptul afişat de interpretorul de comenzi, numărul de parametri şi parametrii comenzii prin care se activează programul, etc.

event-driven = condus de evenimente

Caracteristică a sistemelor de operare de timp real. Momentul planificării pentru execuţie a proceselor este determinat de producerea unui eveniment. Un eveniment poate apare ca efect al tratării unui apel sistem sau provine din mediul exterior. Câteva exemple de evenimente: sfârşitul unei operaţii de intrare/ieşire, modificarea priorităţii unui proces, crearea unui nou proces sau terminarea celui curent, scoaterea unui proces din coada la ceas la momentul cerut.

event flag = indicator de eveniment

Variabilă asociată unui eveniment care indică prin valoarea sa dacă evenimentul s-a produs sau nu. Poate fi de tip boolean sau un bit într-un cuvânt de stare. Unele sisteme de operare oferă apeluri sistem pentru setarea, ştergerea şi testarea variabilelor eveniment. O parte dintre aceste apeluri sunt blocante: procesul se blochează în aşteptarea producerii evenimentului.

exception = excepţie

Situaţie anormală (de obicei o eroare) care poate apare în execuţia unui program. Anumite limbaje (de exemplu Ada) furnizează mecanisme pentru tratarea şi eventual corectarea acestor situaţii, pentru a se evita întreruperea bruscă a unui program. Se ataşează funcţii de tratare a diferitelor excepţii. Chiar dacă aplicaţia se întrerupe, acest lucru se face disciplinat, lăsând structuri de date coerente pe disc, eliberând resursele deţinute, etc.

EXE = EXE

Fişier care conţine un program multisegment în format binar executabil care poate fi recunoscut de încărcătorul (loader-ul) sistemului şi transferat în memoria internă. Din el ia naştere cel puţin un proces gestionat de către sistemul de operare. Unele

sisteme de operare (MS-DOS, VMS, etc.) utilizează extensia **.exe** pentru a desemna astfel de fişiere.

exec = exec

1. Apel sistem în sistemul de operare UNIX prin care se înlocuiesc segmentele procesului apelant prin cele conținute într-un fişier executabil (indicat ca parametru al funcției exec()) şi se începe execuția acestui proces de la punctul său de intrare. Conținutul fişierului executabil înlocuieşte numai imaginea unui porces, nu creează un nou proces. Informația din tabela de procese asociată procesului care a apelat exec() nu este practic modificată.
2. Comandă internă shell care poate primi ca argument o altă comandă (c) şi are ca efect execuția acesteia din urmă în locul shell-ului, fără a fi creat un nou proces.

executive = program executiv

Sinonim mai vechi pentru *sistem de operare* utilizat mai ales pentru calculatoare de talie mare (mainframes). În unele sisteme *executivul* este echivalentul pentru *interpretorul de comenzi* al sistemului de operare (shell).

exit (to) = a termina execuția

În unele sisteme de operare terminarea execuției unui proces se realizează printr-un apel sistem

```
exit()
```

Cu această ocazie se eliberează resursele ocupate, se închid fişierele, etc. Se poate transmite o informație de stare către procesul care a activat procesul care se încheie.

expanded memory = memorie expandată

Schemă hardware proiectată de firmele Lotus, Intel şi Microsoft pentru a permite ataşarea şi utilizarea de blocuri suplimentare de memorie, care să asigure accesul procesorului Intel 8088 la structuri mari de date şi la programe care nu încap în 640Kb. Spațiul virtual de adrese de 1 Moctet al sistemelor IBM-PC este împărțit în 64 de pagini de 16Kb fiecare. Memoria de maxim 32 Mb este împărțită în 2048 pagini de 16 Kb (de fapt *bancuri* sau *blocuri*). Sistemul de gestiune a memoriei expandate permite punerea în corespondență (maparea) celor 64 de pagini pe orice mulțime arbitrar aleasă de blocuri din cele 2048 ale memoriei expandate. Sistemul de memorie expandată EMS (Expanded Memory System) asigură mecanismele hardware pentru mapare şi demapare. Memoria expandată nu trebuie confundată cu memoria extinsă pe care o oferă multe calculatoare personale (memoria instalată după 1Mb). La Intel 80386 şi 80486 memoria extinsă poate fi utilizată pentru a simula memoria expandată în absența unui hardware special. Standardul EMS descrie un mare număr de apeluri sistem pentru a da posibilitatea programelor să mapeze şi să demapeze memorie indiferent de tipul de memorie utilizat (expandată sau extinsă).

extended memory= memorie extinsă

Memoria instalată la adrese peste 1 Moctet pe sisteme IBM-PC al căror procesor permite generarea de adrese mai mari de 220. Intel 80286 poate adresa până la 16Mb, iar 80386 şi 80486 până la 4 Gb.

external fragmentation = fragmentare externă (a memoriei)

Fenomenul de formare a unor zone libere de memorie între segmentele alocate. Aceste zone pot fi alocate dacă sunt suficient de mari, dar se poate ajunge în situația în care zonele libere însumează o cantitate mare de memorie, dar nici una nu este suficient de mare pentru a fi alocată.

Litera F

fast access storage = dispozitiv de memorare (stocare) rapid

Dispozitiv de memorie auxiliară cu timp de acces relativ mic. Se poate stabili dacă dispozitivul de memorare este rapid sau nu în raport cu vitezele relative ale altor dispozitive de memorare din sistem.

fault tolerant (system) = (sistem) tolerant la defecte

Sistem care îşi poate continua execuţia corectă a tuturor funcţiilor sale, (adică execuţia în care rezultatele operării nu conţin erori, iar timpul de execuţie nu depăşeşte limitele specificate), fără o intervenţie din exterior, în prezenţa unei mulţimi specificate de defecte ce pot apărea în timpul funcţionării acestuia. În definirea sistemului tolerant la defecte se presupune că defectările datorate unor erori de proiectare ale sistemului au fost eliminate înaintea începerii utilizării acestuia. Toleranţa la defecte trebuie totuşi să includă şi abilitatea tolerării defectelor de proiectare nedetectate înaintea utilizării sistemului.

FCFS - First Come First Served = primul venit primul servit (FCFS)

Strategie de tratare a cererilor în ordinea sosirii, neţinându-se seama de priorităţi. A se vedea "FIFO".

FIFO - First In, First Out = primul venit primul servit (FIFO)

Strategie şi algoritm de servire a cererilor în ordinea în care sunt primite. Pentru implementare se folosesc structuri de date de tip coadă, reprezentate sub formă de liste la care inserarea se face la sfârşit, iar ştergerile se fac de la început.

file = fişier

Pentru a fi procesate cu ajutorul calculatoarelor, datele sunt organizate în mod structurat, funcţie de aplicaţie şi de relaţiile între ele. Câmpurile de date sunt grupate în înregistrări. O colecţie organizată de înregistrări, memorată sub nume unic, formează un fişier. Deoarece memoriile interne ale calculatoarelor sunt volatile şi au capacitate limitată, fişierele sunt stocate în memorii auxiliare: disc fix, disc flexibil, bandă sau casetă magnetică sau pe un server de fişiere. În UNIX, un fişier este văzut ca o succesiune de octeţi terminată printr-un marcaj de sfârşit de fişier, fără o structură logică de înregistrări; de organizarea logică se ocupă programul de aplicaţii sau biblioteci furnizate în acest scop (de exemplu CISAM pentru fişiere secvenţial indexate utilizabile în Cobol).

File Allocation Table (FAT) = tabelă de alocare a fişierelor (FAT)

În sistemul MS-DOS, cu ajutorul tabelei de alocare a fişierelor se gestionează spaţiul de pe disc (zonele libere, cele alocate şi cele defecte). Plasată pe disc de către programul de formatare, tabela de alocare păstrează lista tuturor unităţilor de alocare ale discului (clustere) care constituie partiţia DOS. Intrările în FAT conţin, după caz, adresa logică a următorului cluster din cadrul fişierului, un indicator de sfârşit de fişier, un indicator de cluster liber sau un indicator de cluster defect. Pentru siguranţă există două copii ale tabelei de alocare a fişierelor pe disc.

file attributes = atributele fişierului

Informaţii asociate unui fişier în unele sisteme de operare pentru a indica modul de acces la fişier (fişierul poate fi citit, scris, executat, utilizat în comun de mai mulţi utilizatori, etc.) sau caracteristici speciale ale acestuia (fişier sistem, fişier invizibil, etc.).

file closing = închiderea fişierului

Totalitatea operaţiilor care se execută atunci când un program termină prelucrarea unui fişier, închizând canalul de acces la informaţiile conţinute în fişier. Dacă se utilizează zone tampon, acestea sunt copiate pe disc. Operaţii viitoare sunt posibile numai după o nouă deschidere a fişierului.

file control block (FCB) = bloc de control al fişierului (FCB)

Structură de date în care sistemul de fişiere din cadrul sistemului de operare memorează o serie de informaţii despre fişier, permiţându-se astfel controlul asupra fişierelor deschise. Prezenţa în memorie a acestei structuri de date creşte viteza de acces prin eliminarea unor adresări costisitoare la disc în vederea obţinerii de informaţii referitoare la fişier şi poziţia lui pe suport extern. În unele sisteme (de exemplu UNIX) există o tabelă alocată static. Intrările în tabelă se asignează la deschiderea unui nou fişier şi se eliberează la închiderea sa, numărul de fişiere deschise simultan de un program fiind limitat de dimensiunile acestei tabele. În alte sisteme, spaţiul pentru FCB se alocă dinamic la deschiderea fişierului şi se include într-o listă înlănţuită a FCB-urilor fişierelor deschise pe un anumit volum.

file directory = catalog

Catalogul este un fişier special ce realizează corespondenţa între numele simbolice ale fişierelor şi identificatorul lor unic în sistem (i-node în UNIX). Prin utilizarea cataloagelor, fişierele se pot grupa după proprietarul lor, aplicaţia la care aparţin, timpul de viaţă în sistem, importanţa lor, etc. Prin realizarea unei ierarhii de cataloage este posibil ca mai multe fişiere distincte să poarte acelaşi nume şi un fişier să fie utilizat cu mai multe nume simbolice. Securitatea fişierelor se realizează şi prin împiedicarea accesării cataloagelor de către utilizatori neautorizaţi. A se vedea "catalogue" şi "directory".

file management = gestiunea fişierelor

Funcţie a unui sistem de operare legată de administrarea informaţiilor organizate şi memorate sub formă de fişiere. Serviciile oferite de un sistem de gestiune a fişierelor sunt: operaţii de bază cu fişiere (localizare după nume simbolic, creare, deschidere, închidere, ştergere fişier), accesul la informaţii în fişiere (citire, scriere, extindere), alocare de spaţiu în mod dinamic pentru fişier pe dispozitive de memorare externă, administrarea spaţiului liber al dispozitivelor de memorie externă, etc. Toate operaţiile legate de stocarea şi regăsirea informaţiilor pe suport extern, de creare a unor canale prin care utilizatorul accesează informaţiile, de utilizare în mod flexibil a acestor canale şi de închidere a lor, organizarea fizică a volumului, protejarea informaţiilor sunt în grija sistemului. Utilizatorul se ocupă numai de prelucrarea acestor informaţii şi de organizarea lor logică (având şi pentru aceasta la dispoziţie mecanisme oferite de sistem). Se pot realiza astfel aplicaţii portabile care nu depind de sistemul de fişiere.

file manager = gestionar de fişiere

1. Program care asigură operaţii cu fişiere cum ar fi: copierea, redenumirea, mutarea şi ştergerea fişierelor. De asemenea, dă posibilitatea creării, suprimării şi examinării cataloagelor aflate pe diferite medii de memorare, realizând şi formatarea discurilor, schimbarea numelor de volume, lansarea în execuţie a programelor. Gestionarul de fişiere este o parte a interfeţei cu utilizatorul care face posibilă nu numai reprezentarea structurii arborescente de fişiere a unui disc, ci şi navigarea prin această structură şi lucrul comod cu fişierele sale.
2. Aplicaţia File Manager este o parte importantă a interfeţei OPEN LOOK ce asigură sistemului de fişiere o interfaţă grafică şi o alternativă la linia de comandă pentru gestiunea cataloagelor şi fişierelor.

file name = nume de fişier

Şir de caractere (alfanumerice) folosite pentru adresarea simbolică a unui fişier. Numărul de caractere din numele fişierului şi tipul acestora este stabilit de către sistemul de gestiune a fişierelor. Numele este o parte din specificatorul de fişier, care în cazul general mai conţine: numele nodului din reţea (maşinii) pe care se află fişierul, numele echipamentului periferic, calea în ierarhia de cataloage de la rădăcină la catalogul în care se află fişierul, extensia, numărul de versiune al fişierului, etc.

file permissions = drepturi/permisiuni de acces la fişier

Operaţii permise unui utilizator asupra unui fişier. Aceste drepturi pot fi: de citire, de scriere, de creare, de ştergere, de execuţie a programului conţinut în fişier, de modificare, etc. Aceste drepturi sunt asociate la crearea fişierelor şi pot fi modificate în concordanţă cu politica de asigurare a securităţii adoptată în sistemul respectiv. Sistemul UNIX împarte utilizatorii în trei categorii în raport cu poziţia lor faţă de un fişier: proprietarul fişierului, utilizatori aparţinând grupului din care face parte utilizatorul şi restul utilizatorilor din sistem (cu excepţia utilizatorului privilegiat numit *root* care poate accesa toate resursele sistemului). Permisiunile de acces se dau pentru aceste trei categorii şi pot fi: citire (r), scriere (w), execuţie (x). În cazul cataloagelor o permisiune x specifică dreptul de a naviga în catalogul respectiv în vederea vizualizării conţinutului său şi alegerii unor fişiere. Permisiunile implicite pot fi limitate printr-o mască de permisiuni.

file server = server de fişiere

Sistem cu disc de dimensiune importantă care oferă servicii de acces la fişiere în reţea. Poate defini zone publice care conţin majoritatea programelor de interes general (de exemplu: Microsoft Word sau Excel, compilatoarele de Pascal sau C++, sisteme de gestiune a bazelor de date, etc), dar şi zone la care accesul este restricţionat, conţinând fişierele proprii ale utilizatorilor. Sistemul de operare permite serverului de fişiere să controleze accesul la resursele comune din sistem, pentru a evita conflictele şi pentru a asigura securitatea şi comunicarea între staţiile care alcătuiesc sistemul. Utilizatorul poate folosi orice staţie de lucru, deoarece fişierele proprii fiecărui utilizator sunt stocate pe serverul de fişiere şi nu pe o maşină anume; indiferent de staţia de la care lucrează el are acces transparent la acelaşi spaţiu de lucru şi la aceleaşi pachete de programe de interes general.

file sharing = partajarea fişierului

Accesarea în comun a fişierelor de către mai mulţi utilizatori la un moment dat.

file signature = semnătură a fişierului

Valoare memorată în fişier ('număr magic") la editarea de legături prin care încărcătorul verifică dacă fişierul conţine cu adevărat un program executabil care poate fi încărcat în memorie şi executat. Permisiunea de execuţie este necesară dar nu şi suficientă pentru aceasta (de exemplu un script shell deşi are permisiuni de execuţie nu poate fi încărcat şi executat; el va fi interpretat de către un shell).

file system = sistem de fişiere

Modul al sistemului de operare având rolul de a administra informaţiile memorate sub formă de fişiere (privite ca resurse). Conţine un mod specific de organizare a informaţiilor pe suport extern şi mecanismele de administrare a acestor informaţii (a se vedea "file management"). Datorită modului diferit de organizare a informaţiilor, un volum creat de un sistem de fişiere nu poate în multe cazuri să fie utilizat în alte sisteme de fişiere.

filter = filtru

Program (utilitar) care face procesarea unui fişier de intrare într-un fişier de ieşire într-un mod specificat prin anumite reguli. Original, a apărut în UNIX, dar acum este folosit şi sub DOS. Utilizarea filtrelor face posibilă construirea de aplicaţii complexe prin conectarea flexibilă între ele a unor programe care îndeplinesc eficient funcţii precise. Aceste programe sunt bine testate, au fost utilizate şi în alte aplicaţii şi cresc productivitatea procesului de dezvoltare de aplicaţii.

find = find

Comandă UNIX pentru găsirea întregii căi până la fişierul al cărui nume este precizat ca parametru şi care respectă anumite condiţii specificate.

finger = finger

Comandă UNIX pentru obţinerea de informaţii despre utilizatorii al căror nume este specificat ca parametru al comenzii. Dacă nu este specificat nici un nume atunci se afişează lista tuturor utilizatorilor cu sesiuni deschise la un moment dat. Poate fi utilizată şi în reţea pentru a obţine informaţii despre utilizatori aflaţi în alte noduri.

flush (to) = a goli (o zonă tampon)

1. Ştergerea unei informaţii nefolositoare sau încheierea unei operaţii.
2. Funcţie pentru golirea zonelor tampon din memorie care au folosit la memorarea unor porţiuni din fişier (deschis pentru scriere sau adăugare).

foreground = prim-plan

1. În sistemele de tip time-sharing, o sarcină în prim-plan este cea capabilă să primească date de intrare de la utilizator şi să întoarcă rezultate la ieşire pentru acesta. În mod normal, există o singură sarcină în prim-plan la un moment dat pentru un terminal (sau fereastră de terminal). Ei îi pot fi asociate însă mai multe procese care lucrează în paralel şi cooperează pentru realizarea sa. În zilele noastre această noţiune este asociată cu UNIX dar ea a apărut la OS/360. A se vedea opusul ("background").
2. Aducerea unei sarcini (*task*) în prim-plan semnifică dorinţa de execuţie a ei imediat, în timp ce ducerea ei în fundal echivalează cu o scădere a priorităţii de rulare (aceasta se rulează numai când nu mai sunt alte sarcini în prim-plan).

fork = fork

Apel sistem UNIX pentru generarea unui nou proces (proces identic cu procesul apelant). Procesul care apelează funcţia este numit *părinte* iar cel creat este numit *fiu*. Întreaga imagine a procesului *părinte* este copiată pentru a crea procesul *fiu*, cu

excepția segmentului de cod pur care nu se duplică ci se utilizează în comun. Procesul *fiu* moştenește întregul context de execuție și fișierele deschise. Din acest moment cele două procese se separă, dar ele continuă să interacționeze prin anumite mecanisme.

format (to) = a formata

A se vedea "disk formatting".

fragmentation = fragmentare (a memoriei)

Împărțirea memoriei disponibile în zone necontigue, în fragmente, a căror gestiune și alocare se face mai dificil decât în cazul existenței unei zone unice de memorie liberă. Fenomenul conduce la pierderi de memorie datorită faptului că unele zone nu se mai pot aloca. A se vedea "external fragmentation" și "internal fragmentation".

Frequently Asked Questions (FAQ) = întrebări puse frecvent (FAQ)

Documente care conțin liste de întrebări apărute frecvent pe un anumit subiect și răspunsuri la ele. Administratorii de sistem folosesc acest instrument pentru a beneficia de experiența altor administratori care se ocupă de mai mult timp de un sistem similar și au întâlnit aceleași probleme. Multe grupuri de știri și-au construit documente FAQ voluminoase care încearcă să răspundă tuturor întrebărilor pe care le-ar putea pune utilizatorii.

Litera G

garbage collection = colectarea (refacerea) spațiului disponibil

Termen care desemnează o clasă de strategii pentru realocarea dinamică, transparentă a memoriei (fără a cere ca dealocarea și alocarea să se facă explicit de către programe de nivel înalt). Conceptul și terminologia provin din implementările limbajului LISP, dar și alte limbaje care oferă alocare dinamică folosesc colectarea spațiului disponibil. În sisteme de operare se utilizează pentru a desemna procedurile prin care se realizează defragmentarea memoriei. De exemplu se poate face compactarea memoriei prin mutarea zonelor alocate spre adrese mici ale memoriei, ceea ce conduce la deplasarea zonelor libere spre adrese mari și formarea unei zone libere de dimensiuni importante. Există și alți algoritmi mai eficienți. Se aplică atât memoriei interne cât și memoriilor auxiliare.

GCOS (General Comprehensive Operating System) = GCOS

Sistem de operare construit de firma General Electric sub numele GECOS (the General Electric Comprehensive Operating System); a evoluat din System/360 DOS. Mai târziu i s-au asociat primitive de time-sharing și procesarea tranzacțiilor. Numele a fost schimbat în GCOS (General Comprehensive Operating System) când firma Honeywell a cumpărat divizia de calculatoare a GE. Sistemul GECOS/GCOS a influențat dezvoltarea sistemului UNIX. Unele sisteme UNIX de la Bell Labs au folosit mașinile GCOS pentru virtualizarea imprimantei și alte servicii; câmpul adăugat la fișierul '/etc/passwd' pentru informația de identificare GCOS a fost numit *câmp GCOS*.

GECOS (the General Electric Comprehensive Operating System) = GECOS

A se vedea "GCOS".

GNU = GNU

Pachete de programe compatibile UNIX realizate în cadrul unui proiect al FSF (Free Software Foundation), proiect condus de Richard Stallman. Ideea de bază a proiectului GNU este că informația este proprietatea comunității și că sursele trebuie să fie publice. GNU EMACS și compilatorul GNU C, două realizări ale acestui proiect sunt foarte cunoscute și folosite în prezent. GNU este acronim pentru **G**NU 's **N**ot **U**NIX

grep = grep

Comandă UNIX (filtru) pentru scanarea rapidă a unui fișier sau a unui set de fișiere și trimiterea la fișierul standard de ieșire

a liniilor care corespund unui şablon (o expresie regulată). Instrument deosebit de util în activitatea de administrare de sistem. Numele provine din comanda *g/re/p* specifică editorului qed/ed, *re* având semnificaţia de expresie regulată (**r**egular **e**xpression) sau poate fi considerat un acronim pentru **G**lobal **R**egular **E**xpression **P**rint.

Litera H

hacker = spărgător de cod

1. Persoană capabilă să acceseze informaţii (nu neapărat în sens distructiv) în sisteme de prelucrare a informaţiilor la care nu are în mod normal drepturi de acces.
2. Persoană care explorează detaliile de implementare şi de utilizare ale sistemelor în vederea utilizării extensive a tuturor facilităţilor acestora, spre deosebire de majoritatea utilizatorilor care preferă să înveţe minimul necesar.
3. Persoană care scrie şi implementează programe cu entuziasm, dovedind calităţi deosebite de programator. Aspectele teoretice sunt de obicei lăsate pe plan secundar. Se trece foarte repede la implementare fără a teoretiza prea mult.
4. Expert în utilizarea unui program sau sistem (de exemplu "UNIX hacker").
5. În sens peiorativ se foloseşte pentru persoane care-şi folosesc cunoştinţele tehnice pentru a detecta slăbiciuni ale sistemelor de protecţie şi a reuşi pătrunderea neautorizată (accesul neautorizat) în sistemele private de calculatoare şi în băncile de date. Denumirea corectă în acest caz este "cracker".
6. Termenul de "hacker" tinde să desemneze şi apartenenţa la o anumită comunitate, definită în cadrul reţelelor de calculatoare, care respectă o etică a spărgătorilor de cod.

Termenul "hacker" presupune în fapt o îmbinare a mai multora dintre caracteristicile descrise mai sus.

hacker ethic = etica spărgătorilor de cod

Ansamblu de principii acceptate şi respectate de o comunitate din cadrul reţelei de calculatoare:

1. Atitudine pozitivă faţă de partajarea informaţiilor. Experienţa se pune la dispoziţia celorlalţi prin scrierea de programe la care accesul este liber; se facilitează accesul la informaţii şi la resursele de calcul disponibile.
2. Accesul neautorizat într-un sistem pentru amuzament sau explorare de informaţii este acceptat, atâta timp cât "spărgătorul de cod" nu distruge sau divulgă informaţii confidenţiale.

Primul principiu este acceptat de majoritatea hacker-ilor; mulţi dintre ei scriu programe la care accesul este liber. Filozofia pe care se bazează proiectul GNU merge mai departe şi consideră că accesul la *absolut toate* informaţiile trebuie să fie liber.

Principiul 2 este controversat, unele persoane considerând că accesul la informaţiile private nu este etic. Alţi spărgători de cod consideră că rolul lor este benefic: după ce pătrund într-un sistem, fără să distrugă informaţii, dau dovadă de curtoazie explicând administratorului de sistem printr-un e-mail cum au reuşit acest lucru. În felul acesta slăbiciunile sistemului de securitate pot fi cunoscute şi remediate, prevenind viitoare atacuri distructive.

Cea mai importantă caracteristică a hacker-ilor constă în dorinţa acestora de a partaja cunoştinţele tehnice, programele şi resursele de calcul. Reţele imense precum USENET, FidoNet şi Internet funcţionează fără un control centralizat, bazându-se pe cooperare.

halt (to) = a opri (definitiv)

1. La nivelul unui sistem de operare, procedură de oprire a sistemului la dorinţa administratorului (operatorului) sau la întâlnirea unei situaţii grave de eroare (a se vedea şi "crash"). Dacă sistemul are mai mulţi utilizatori, numai utilizatorul privilegiat poate executa această procedură. După închiderea tuturor sesiunilor şi eliberarea tuturor resurselor, se execută instrucţiunea **halt** care face parte de regulă din categoria instrucţiunilor privilegiate, executabile numai în mod sistem (kernel). Ea opreşte efectiv maşina şi o trece într-o stare specială de unde se va putea realiza încărcarea unui alt sistem de operare sau întreruperea alimentării.
2. În anumite limbaje de programare, instrucţiunea "halt" opreşte execuţia unui program în situaţia întâlnirii sfârşitului logic al programului sau a unei situaţii de eroare. Se dă controlul mediului sub care se execută programul (de exemplu Turbo Pascal) sau sistemului de operare pentru execuţia unei noi comenzi.

handler = rutină de tratare

Funcție asociată unui eveniment sau clasă de evenimente. Când se produce evenimentul, sau un eveniment din clasă, funcția este executată.

hang = a (se) bloca, a rămâne agățat

1. A aştepta producerea unui eveniment care nu apare niciodată (de exemplu sistemul aşteaptă terminarea unei citiri de pe discul sistem, dar acesta a căzut în pană).
2. A aştepta un anumit eveniment până când se întâmplă ceva. De exemplu, în cazul afişării unui meniu pe ecran se aşteaptă apăsarea unei taste.

Starea de blocare definită de "hang" este diferită de starea definită prin "crash" în care programul sau sistemul este de asemenea neutilizabil (dar deoarece nu mai funcționează şi nu pentru că aşteaptă producerea unui eveniment). Recuperarea din ambele situații se realizează deseori la fel.

hardware configuration = configurație (a echipamentelor)

Totalitatea resurselor fizice (permanente) ale unui sistem şi modul în care acestea sunt conectate între ele. Dintre aceste resurse pot face parte: procesoare (este important cum sunt interconectate, la ce memorii au acces, cum schimbă informații), memorii (interesează câte module există, ce capacitate are fiecare, ce mecanisme de gestiune sunt oferite prin hardware), echipamente periferice (discuri, benzi magnetice, cum sunt conectate la cuploare, cum fac acces la memorie, ce adrese au şi cum produc întreruperi), interfețe de rețea, etc. Sistemul de operare trebuie să gestioneze aceste resurse, deci configurația la un moment dat trebuie să fie corect reflectată în structurile de date ale sistemului. Unele sisteme de operare nu permit modificarea configurației în timpul executării lor. La astfel de sisteme, adăugarea unui nou disc presupune oprirea sistemului, reconfigurarea nucleului şi repornirea sistemului care gestionează o nouă configurație. Anumite aplicații au de suferit dacă se opreşte sistemul. Tehnicile moderne "plug and play" tind să soluționeze această problemă.

head = head

Comandă UNIX de tip filtru pentru afişarea la ieşire a primelor *n* linii din fişierul specificat în linia de comandă (împreună cu valoarea lui *n*). Dacă acest număr de linii nu este specificat, valorea sa implicită este 10.

headway = fracție de timp

În cazul sistemelor cu multiprogramare, fracțiunea din timpul total utilizată efectiv de fiecare dintre lucrările planificate pentru execuție.

heap = zonă de memorie pentru alocare dinamică

1. Zonă de memorie folosită pentru a aloca la cerere, în cursul execuției programului, spațiul de memorie necesar unor structuri de date. Aceste spații pot fi eliberate tot la cerere şi refolosite. Zona este structurată şi gestionată prin funcții specifice, incorporate în program la editarea de legături.
2. Structură de date ("movilă") utilă în implementarea eficientă a cozilor bazate pe priorități, a unor algoritmi de sortare şi a altor mecanisme de prelucrare rapidă a informațiilor.

help = help

Sistem de comenzi sau comandă pentru afişarea de informații utile despre diverse comezi ale unui sistem sau program de aplicație. În sistemul UNIX această comandă consumă mai puține resurse decât comanda "man", dar informațiile pe care le oferă sunt mai sărace.

hidden = invizibil (despre fişiere)

Fişierele cu acest atribut nu sunt vizibile în cazul listării cu comenzi obişnuite a componentelor unui catalog. Ascunderea fişierelor se face pentru a limita posibilitatea de ştergere a acestora, sau din motive de securitate a datelor.

hierarchical = ierarhic

Atribut asociat unor elemente dintr-un sistem, legat de modul în care sunt structurate acestea. De exemplu un sistem de fişiere poate avea o structură ierarhică arborescentă, compusă din cataloage şi fişiere. Un sistem de operare poate fi construit ca o ierarhie de module, modulele plasate mai jos oferind servicii celor plasate deasupra lor. O ierarhie poate fi "slabă" (în cazul în care se pot cere direct servicii de la orice nivel) sau "strânsă" (în acest caz toate cererile se adresează nivelului celui mai înalt, fiecare modul putând să ceară servicii numai de la nivelul plasat imediat sub el).

High Memory Area (HMA) = zonă de memorie înaltă

Zonă de memorie cu capacitatea de 64 Kb situată imediat după primul 1Mb de memorie la calculatoare personale. Procesoarele Intel 8086/8088 puteau accesa numai 1 Mb de memorie, pe când 80286 putea accesa 16 Mb, iar 80386 până la 4 Gb. Din motive de compatibilitate "înapoi", procesoarele 80286 şi 80386 au fost proiectate să opereze şi în "mod real", un mod în care ele se comportă exact ca 8086/8088, putând accesa 1 Mb de memorie şi în plus această memorie suplimentară de până la 64 Kocteți, care este HMA.

hit = găsire (a unei informații)

Găsirea informațiilor dorite în memoria rapidă (dar de capacitate mică) în cazul utilizărrii unei memorii organizate ierarhic (de exemplu în memoria cache). În acest caz accesul la informația dorită se realizează mai rapid decât în cazul în care aceasta ar trebui adusă din memoria mai lentă (dar de capacitate mai mare).

hit ratio = frecvența succeselor în găsirea informațiilor

Raportul dintre numărul de adresări care găsesc informația în memoria rapidă şi numărul total de adresări. Depinde de comportamentul programelor (cât de mult îşi concentrează adresările în spațiul de memorie sau în timp) şi de mărimea memoriei rapide. Dacă se măreşte memoria intermediară, probabilitatea de succes este mai mare, numărul de adresări la memoria principală scade, ceea ce conduce la un trafic redus pe magistrală, permițând conectarea mai multor procesoare în sistem (în cazul unui sistem distribuit).

hold and wait = alocare parțială ("păstrează şi aşteaptă")

Tehnică de alocare a resurselor prin care o lucrare (job) primeşte resursele pe rând, pe măsură ce le solicită în timpul execuției. Dacă resursele nu sunt disponibile la un moment dat, se aşteaptă eliberarea lor de către lucrarea care le deține (fără a elibera resursele deja acaparate). Este una dintre condițiile necesare pentru producerea blocajului reciproc într-un sistem. O alternativă este "alocarea totală": programul nu este lansat în execuție până nu sunt disponibile toate resursele pe care le solicită.

hot key = tastă cu funcție specială

Mecanism prin care în sistemul MS-DOS se activează un program TSR la apăsarea unei taste (combinație de taste) căreia i s-a atribuit această funcție specială. Un program TSR (Terminate and Stay Resident) ocupă memorie, dar nu este planificat pentru execuție de către sistem. El primeşte controlul prin capturarea şi "supravegherea" întreruperilor de la tastatură. Când este primit un caracter, dacă acesta nu are o semnificație specială el este pus într-o coadă a sistemului de operare, iar TSR revine din întrerupere în programul în curs de execuție. Dacă este un caracter căruia i s-a asociat o funcție specială, atunci se execută secvența corespunzătoare din TSR.

HPFS High-Performance Filing System = HPFS

Sistem de memorare a fişierelor pe disc pentru sistemul de operare OS/2. HPFS este o metodă pentru reținerea locațiilor ocupate de toate fişierele din fiecare catalog. A fost introdus începând cu versiunea 1.2 a sistemului de operare OS/2. Este mai eficient decât sistemul FAT pentru DOS. HPFS acceptă nume de fişiere lungi, fără restricția la maxim 8 caractere a numelui fişierului şi la 3 caractere a extensiei acestuia. Pentru mărirea performanțelor, HPFS exploatează structuri de date pentru diferite nivele de memorie intermediară. HPFS permite asocierea atributelor extinse EAS (Extended Attributes) fişierelor şi directoarelor.

HP-UX = HP-UX

Versiune de UNIX care rulează pe stațiile de lucru din familia Hewlett-Packard 9000.

Litera I

i-node = i-node

1. În sistemul de fişiere al sistemelor de operare din familia UNIX, "i_node" este un bloc de pe disc care păstrează informaţii despre un fişier. Există mai multe tipuri de sisteme de fişiere. La cele descrise în literatura de specialitate, în "i_node" pot fi memorate: tipul fişierului, numărul de legături spre el, identitatea proprietarului şi a grupului, permisiunile de acces la fişier, informaţii de regăsire a blocurilor alocate fişierului, dimensiunea în octeţi a fişierului, data şi ora ultimului acces, data şi ora ultimei modificări a fişierului, momentul creării, etc. Pentru regăsirea blocurilor de date ale fişierului sunt memorate adresele a 13 blocuri de disc: 10 blocuri directe şi 3 blocuri indirecte. Pentru fişierele mici, care conţin cel mult 10 blocuri, adresele acestor blocuri se află direct din i_node. Pentru fişierele mai mari se utilizează şi primul bloc indirect, numit *bloc indirect simplu* care indică adresa unui bloc de pe disc conţinând la rândul lui adresele a 128..256 blocuri de date. Pentru fişierele deosebit de mari se folosesc în acelaşi mod şi blocurile 12 şi 13, numite *bloc indirect dublu* şi respectiv *bloc indirect triplu*. Cu cât fişierul este mai mare, cu atât informaţiile plasate spre sfârşitul său necestă un timp de acces mai mare, datorită multiplelor adresări la disc trecând prin blocuri indirecte. Pentru acces rapid la informaţii, la deschiderea fişierului toată informaţia conţinută în i-node este adusă de pe disc în memoria principală într-o structură de date (o intrare în tabela de i_node-uri).
2. În sistemele de operare din familia UNIX, "i_node" este identificatorul unic pe care orice fişier îl primeşte în sistemul de fişiere. Un fişier poate avea mai multe nume simbolice (link-uri), poate fi deschis simultan de mai multe procese, dar el este în mod unic determinat prin acest număr întreg care are legătură cu poziţia fişierului respectiv în sistemul de fişiere.

I/O buffer = tampon de intrare/ieşire

Pentru sporirea vitezei operaţiilor de intrare/ieşire, modulul de administrare a echipamentelor periferice utilizează zone tampon. Sunt folosite tehnici cum ar fi "double-buffering", "triple-buffering", "multiple-buffering". Tamponul de intrare/ieşire utilizat pentru memorarea informaţiilor provenite de la echipamentele periferice sau destinate acestora este o zonă de memorie rezervată în spaţiul de adrese al programului executabil sau într-un spaţiu pus special la dispoziţie de către sistemul de operare (*zona dinamică*). Aceste "buffere" pot fi utilizate şi ca o memorie tampon (*cache*) menită să accelereze accesul la informaţii în mod transparent faţă de utilizator. Sistemul de operare UNIX menţine o mulţime de astfel de zone tampon pe care le alocă dinamic proceselor care execută operaţii de intrare/ieşire.

IBM (International Business Machines) = IBM

Firmă americană, cel mai mare producător de calculatoare din lume. Printre calculatoarele intrate în istorie produse de această firmă s-au numărat IBM 1401, IBM 7040, IBM 7090, seriile 360 şi 370 (din gama calculatoarelor "mainframes"). Cercetarea desfăşurată de IBM a adus multe inovaţii în domeniul arhitecturii calculatoarelor, în domeniul sistemelor de operare (tehnici de memorie virtuală, sisteme "batch" şi "time-sharing", organizări de fişiere ISAM, etc), al limbajelor (este suficient să fie amintită numai crearea limbajului Fortran), al bazelor de date, al instruirii asistate de calculator, ş.a.m.d. IBM a fost iniţiatorul producerii de calculatoare personale. Prima generaţie de calculatoare IBM-PC a apărut în anul 1981, cu procesor pe 16 biţi produs de firma INTEL. S-au dezvoltat apoi modele bazate pe microprocesoare 80286, 80386, 80486 şi PENTIUM care respectă compatibilitatea în jos cu calculatoarele produse anterior. În domeniul staţiilor de lucru, ca şi al sistemelor multiprocesor cu procesare simetrica, IBM produce sisteme performante bazate pe procesoare risc RS6000.

IBM PC compatible computer = calculator compatibil IBM-PC

PC este acronimul pentru **P**ersonal **C**omputer (calculator personal). Dacă acest PC nu este produs de către IBM, ci de către o altă firmă, dar respectând standardul specific IBM, el se numeşte *"calculator compatibil IBM-PC", "calculator compatibil PC", "compatibil PC",* sau simplu *"PC".* Compatibilitatea este atât la nivelul arhitecturii cât şi la nivelul programelor.

id = identificator unic

1. În sistemele care implică gestiunea resurselor pot exista mecanisme şi servicii de asociere de nume simbolice diferitelor obiecte (fişiere, hosturi, utilizatori, procese, etc). Sistemele de operare, atât cele concentrate cât şi cele de reţea sau distribuite gestionează însă mai comod resursele dacă acestea sunt identificate prin numere de lungime fixă în loc de nume simbolice complicate (şiruri de caractere de lungime variabilă având sau nu câmpuri cu informaţii specifice). Identificatorul unic asociat unui obiect este în sistemele UNIX un număr întreg fără semn.. În anumite sisteme de fişiere distribuite, acest identificator este o informaţie numerică mai complicată, putând conţine informaţii

codificate pentru localizarea fişierului în sistemul distribuit.
2. Comandă UNIX pentru afişarea de informaţii de identificare a utilizatorului (număr şi nume pentru utilizator şi grup). Exemplu:

```
$ id

uid = 31106(mdobre) gid = 4001(profs)
```

idempotent = idempotent

1. Caracteristică a unui operator, care chiar dacă este utilizat în mod repetat, efectul coincide cu cel al utilizării lui o singură dată. Termenul se foloseşte pentru subprograme (funcţii, proceduri) care trebuie să execute acţiuni critice exact o singură dată, chiar dacă rutina este apelată de mai multe ori. Proprietatea este importantă în cazul mecanismului de apel de proceduri la distanţă (RPC). Dacă *serverul* se defectează sau există o eroare în sistemul de comunicaţii care face ca rezultatul executării apelului la distanţă să nu poată ajunge la *client*, după un timp acesta va detecta *time-out* şi eventual va repeta apelul. Serverul va fi în situaţia de a repeta execuţia procedurii (în cazul defectării serverului, după ce acesta va reporni). Există operaţii care sunt idempotente prin natura lor: de exemplu inserarea unui nou element într-un şir ordonat strict crescător, astfel încât şirul să rămână ordonat strict crescător (dacă elementul aparţine deja şirului, el nu se mai inserează). În orice ordine s-ar insera elementele, indiferent de câte ori se repetă introducerea oricărui element, rezultatul va fi acelaşi. Termenul este frecvent folosit cu referire la fişierele antet (.h), care conţin definiţii şi declaraţii comune de inclus în mai multe fişiere sursă C. Dacă la o aceeaşi compilare, un fişier este inclus de mai multe ori (de exemplu prin fişiere "`#include`" imbricate), pot apare erori de compilare în cazul în care acest fişier inclus nu se protejează singur la incluziunile multiple. Un astfel de fişier protejat este numit *idempotent*.
2. Din punct de vedere matematic, o funcţie f : D->D este idempotentă, dacă f(f(x)) = f(x), pentru orice x în D, adică aplicarea ei repetitivă are acelaşi efect cu aplicarea ei o singură dată. Acest lucru poate fi extins şi pentru funcţiile cu mai multe argumente (de exemplu operatorul boolean & are proprietatea x & x = x). Fiecare valoare a imaginii unei funcţii idempotente este un *punct fix* al funcţiei.

idle = liber, neutilizat

Starea unei resurse care nu este solicitată de nici un proces la un moment dat (de exemplu starea procesorului când nu există nici un proces *pregătit pentru execuţie*). În sistemele de calcul moderne procesorul nu poate fi oprit în situaţia în care nu este solicitat de nimeni (o explicaţie pentru această decizie putând fi aceea că există acţiuni asincrone care nu sunt planificate pentru execuţie de către planificatorul de procese). O abordare posibilă este aceea de a planifica în acest caz pentru execuţie un proces special (numit în unele sisteme de operare "idle process"). Acest proces nu aparţine nici unui utilizator şi nici nu consumă resurse. El pur şi simplu nu face nimic, aşteptând să apară un proces executabil. Procesul special poate fi un ciclu pe o instrucţiune NOP în aşteptarea unei întreruperi sau poate fi repetarea ciclică a apelului sistem care forţează planificarea pentru execuţie, având ca rezultat replanificarea procesului special atâta timp cât nu apar alte procese.

IEEE (Institute of Electrical and Electronics Engineers) = IEEE

Organizaţie a inginerilor din domeniul electric (electrotehnică, energetică, electronică, tehnică de calcul), cu sediul central în SUA, dar care acceptă membri din toată lumea, organizând *chapter-e* naţionale grupate în *regiuni* pe criterii geografice. Organizaţia promovează interesele inginerilor şi impulsionează cercetarea ştiinţifică şi răspândirea cunoştinţelor din domeniile enumerate prin acţiuni diverse cum ar fi: publicarea de reviste de specialitate de înaltă ţinută ştiinţifică, organizarea, sponsorizarea şi sprijinirea de manifestări ştiinţifice, organizarea unei informări ample asupra locurilor de muncă disponibile în domeniu, distribuirea de cărţi, cataloage, produse multimedia, etc. O activitate importantă este cea de standardizare. Sunt cunoscute standardele din domeniul reţelelor, interfeţelor, magistralelor, apelurilor sistem UNIX, etc. Avantajul major al acestor standarde este acela că ele nu reprezintă interesele vreunui producător sau firme.

indexed file = fişier indexat

Fişier la care înregistrările se accesează pe baza unor chei (primare sau secundare). Pentru regăsirea rapidă a informaţiilor în cazul unor volume mari de date, pentru fiecare înregistrare se definesc unul sau mai multe câmpuri după care se face căutarea, numite *chei*. Nu se mai parcurg toate informaţiile (citirea lor de pe suport extern este o operaţie costisitoare ca timp), ci numai cheile. Acestea sunt ordonate, ceea ce permite utilizarea unor algoritmi eficienţi de căutare. Împreună cu fiecare cheie se păstrează un pointer către zona din fişier care conţine înregistrarea asociată. În cazul bazelor de date mari,

căutarea într-un astfel de fişier este mult mai eficientă decât căutarea secvenţială.

indexed-sequential file = fişier secvenţial-indexat

Fişier indexat cu cheie primară, adică ordonat după această cheie primară. După localizarea acesteia, căutarea se face secvenţial, dar într-o zonă limitată a fişierului. O metodă de organizare a acestor fişiere este ISAM, care a fost introdusă de către IBM pentru Cobol şi se aplică cu succes şi în prezent (a se vedea "indexed-sequential access method").

indexed-sequential access method (ISAM) = metodă de acces secvenţial-indexat (ISAM)

Procedură de memorare şi regăsire a informaţiilor într-un fişier aflat pe disc. Se creează o mulţime de indecşi pentru a descrie unde sunt localizate pe disc înregistrările fişierului sau grupuri de înregistrări. Aceşti indecşi se pot memora într-un fişier index separat. Folosind o astfel de procedură, se obţine o localizare rapidă a înregistrărilor eliminând citirile inutile ale tuturor datelor de la început până la informaţia dorită. Pentru limitarea spaţiului de căutare, fişierul index poate fi structurat în două sau mai multe fişiere, de exemplu un index primar şi unul secundar (corespunzând cheilor primare şi respectiv secundare). Această organizare implică efectuarea mai multor accesuri la disc: se accesează întâi fişierul primar pentru a se localiza zona în care se caută cheia în fişierul secundar, apoi fişierul secundar pentru a localiza zona în care se caută informaţia în fişierul de date şi în final se realizează o căutare secvenţială într-o zonă limitată din fişierul de date. Accelerarea căutării se poate realiza prin utilizarea algoritmilor de dispersie (*hash-coding*) şi a altor metode. Accesul pur secvenţial este de asemenea posibil, dacă sunt necesare prelucrări în care se parcurg în serie toate înregistrările din fişier sau dintr-o secţiune a lui. Un exemplu intuitiv de fişier secvenţial-indexat îl oferă dicţionarul. Utilizând litera cu care începe cuvântul căutat se localizează zona în care sunt păstrate cuvintele care încep cu acea literă. Apoi prin căutare secvenţială se localizează două cuvinte (sau silabele lor de început) între care se află situat din punct de vedere lexicografic cuvântul căutat. În felul acesta se reduce căutarea cuvântului la cel mult două pagini de dicţionar. Dacă cuvântul nu există în dicţionar, căutarea se opreşte la primul cuvânt "mai mare" decât el (în ordinea lexicografică).

input-output bound = limitat în intrare/ieşire

Comportament al unui program, proces sau lucrare care utilizează cea mai mare parte a timpului pentru execuţia de operaţii de intrare/ieşire (intuitiv, timpul lui de răspuns este limitat de viteza cu care se fac operaţiile de intrare/ieşire), spre deosebire de programele "limitate unitate centrală" care utilizează intens procesorul şi ale căror performanţe depind de viteza şi disponibilitatea acestuia. Un echilibru al sistemului şi o bună utilizare a resurselor se obţine prin mixarea corespunzătoare a unor programe din ambele categorii.

interactive mode = mod interactiv (conversaţional)

Mod de acces la serviciile unui sistem de calcul care asigură o comunicaţie directă a utilizatorului cu calculatorul prin intermediul unui terminal de intrare/ieşire. Până la apariţia terminalelor cu tub catodic, strategia preferată în exploatarea unui sistem de calcul era aceea de a-l îndepărta pe utilizator de sistem, pentru ca, prin planificarea corespnzătoare a execuţiei programelor să se obţină performanţe cât mai ridicate. Se încerca limitarea chiar a intervenţiilor operatorului pentru ca sistemul să înlănţuie automat prelucrările şi să ia decizii cu o viteză superioară celei specifice agentului uman. Acestea au fost sistemele de operare cu organizare "pe loturi".

interactive processing = prelucrare interactivă

Prelucrarea controlată de către utilizator de la un terminal.

interactive program = program interactiv

Program care în timpul rulării sale, realizează un dialog cu utilizatorul. Acesta introduce datele la cererea programului şi ia decizii de care depinde fluxul controlului.

interactive system = sistem interactiv

Sistem în care utilizatorul lucrează în mod interactiv (interacţionează cu sistemul prin intermediul unui terminal).

interface = interfaţă

Totalitatea regulilor, convențiilor, modulelor program şi dispozitivelor cu ajutorul cărora un element al unui sistem comunică cu un alt element al sistemului sau cu utilizatorul. În cazul programelor mari, modularitatea este o soluție pentru proiectarea, implementarea şi întreținerea eficientă a aplicației respective. Dacă se definesc corect interfețele dintre module şi apelurile între module se fac corect, modificarea unor module nu va afecta funcționarea corectă a celorlalte şi a ansamblului şi nu va impune testarea sau modificarea altor componente ale sistemului.

Una dintre sarcinile importante ale unui sistem de operare (şi a oricărei aplicații) estre aceea de a-i oferi utilizatorului o interfață cât mai comodă cu sistemul. Această interfață se realizează la mai multe niveluri:

a. interfața cu utilizatorul prin limbajul de comenzi
b. interfața cu programele utilizatorului

a) Interfața cu utilizatorul presupune existența unui interpretor de comenzi. În sistemele UNIX acesta se numeşte *shell*. El oferă şi facilități deosebite de programare, ceea ce face ca unele aplicații să se realizeze mai comod la acest nivel decât într-un limbaj de programare cum ar fi C. Există interfețe alfanumerice, interfețe grafice orientate pe menu-uri şi ferestre, interfețe "inteligente" bazate pe recunoaşterea vocii sau a semnăturii, etc. Utilizarea unora dintre ele implică existența unor terminale cu caracteristici deosebite.

b) Programele utilizatorilor solicită servicii sistemului de operare prin intermediul apelurilor sistem. Aceasta presupune existența pe de o parte a unor mecanisme prin care se dă controlul sistemului, iar pe de altă parte existența unei interfețe specifice fiecărui limbaj.

interleaved = întrețesut (despre execuție)

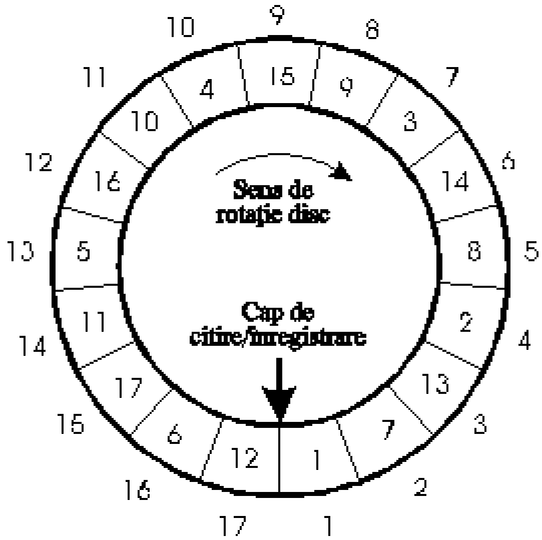
Politică de planificare a proceselor pentru execuție bazată pe comutarea contextului. După ce un proces a utilizat un timp procesorul, un alt proces primeşte această resursă, urmând ca primul să revină după un timp. Această planificare poate fi bazată pe cuante de timp, dar întrețeserea execuției poate fi controlată şi de evenimente. Tehnica este cunoscută şi sub numele de *multiplexarea procesorului*.

interleaving = întrețesere (despre blocuri pe disc)

Tehnică de dispunere a blocurilor pe disc care asigură scăderea timpului de acces la informații. Discurile nu sunt dispozitive start-stop care să poată fi oprite un timp nedefinit înaintea citirii unui bloc (sector). Atunci când se prelucrează toate blocurile unei piste (sau mai multe blocuri succesive) este posibil ca blocul $k+1$ să ajungă sub capul de citire înainte ca sistemul să aibă timpul necesar să prelucreze (sau să mute din buffer) informația citită din blocul k. Pentru accesarea blocului $k+1$ este necesară încă o rotație de disc. Dacă se "întreţes" însă blocurile logice cu un factor de întrețesere n, blocul logic $k+1$ va fi plasat într-un bloc fizic distanțat la n blocuri de cel în care este memorat blocul logic k. Timpul de rotație necesar pentru ca cele n blocuri să treacă pe sub capul de citire va asigura un timp suplimentar în care informația din blocul logic k să fie prelucrată (*timp critic*).

În exemplul din figura alăturată este reprezentată o pistă a unui disc cu 17 sectoare/pistă. În exterior sunt indicate adresele fizice, iar în interior cele logice.

a. fără întrețesere: este posibil ca prelucrarea celor 17 sectoare să necesite 17 rotații ale discului;
b. cu factor de întrețesere 3: presupunând că prelucrarea unui bloc necesită un timp inferior celui critic, toate cele 17 blocuri pot fi prelucrate în trei rotații ale discului.



internal fragmentation = fragmentare internă

Fenomen care constă din neutilizarea unei părţi din ultima unitate de memorie alocată unui program. Fiind zone interne ale unor programe, aceste spaţii nu mai pot fi folosite.

În cazul administrării memoriei cu partiţii specificate static (MFT), spaţiul neutilizat este la sfârşitul partiţiei.

În cazul paginării, deoarece paginile au lungime fixă iar spaţiul de adrese nu are o dimensiune multiplu de dimensiunea paginii, se alocă un număr întreg de pagini ceea ce conduce la neocuparea completă a spaţiului din ultima pagină. Spaţiul pierdut este egal în medie cu jumătate din lungimea paginii (din acest punct de vedere este indicată utilizarea de pagini de dimensiuni mici).

0	Sistem de operare
64Kb	Lucrare 1 / Proces 1
	Neutilizat
256Kb	Lucrare 2 / Proces 2
	Neutilizat
1Mb	. . .
7Mb	Lucrare n / Proces n

	Neutilizat

interprocess communication (IPC) = comunicație între procese

Mijloace prin care procesele schimbă informații între ele şi se sincronizează. Se constată tendința realizării de aplicații modulare, structurate în mai multe procese care comunică între ele, fiecare îndeplinind o funcție simplă. În felul acesta se favorizează paralelismul, şi se disciplinează ingineria programelor (este mai uşoară proiectarea, implementarea sau demonstrarea corectitudinii unor programe simple, decât a unora complexe). Procesele pot comunica în două moduri fundamentale: schimb de mesaje şi acces la zone comune de memorie. Arhitectura sistemului joacă un rol important în alegerea unuia dintre cele două moduri. Sistemul UNIX oferă mai multe mecanisme de comunicație între procese. Dintre acestea: cozile de mesaje, semafoarele şi zonele de memorie partajată au fost introduse de către versiunea UNIX System V şi sunt cunoscute sub numele de *IPC System V*. Ele au o tratare unitară în sistem, fiind accesibile utilizatorului prin apeluri sistem.

interrupt = întrerupere

1. Eveniment care întrerupe (temporar) fluxul normal de execuție a instrucțiunilor şi activează o rutină de tratare a întreruperii. Acesta este mecanismul fundamental prin care maşina ia cunoştință de evenimente produse asincron în raport cu funcționarea sa (în lumea exterioară sau în alte părți componente ale sistemului). Pe baza sa, sistemul de operare poate implementa şi controla concurența proceselor, multiplexarea utilizării unor resurse, suprapunerea în timp a unor activități (cum ar fi cele de calcul cu cele de intrare/ieşire, etc.). În majoritatea sistemelor de operare, pentru executarea unui *apel sistem* se dă controlul nucleului utilizând întreruperi produse sincron prin execuția unor instrucțiuni special prevăzute (trap-uri, derutări, SVC, etc).
2. În MS-DOS termenul *întrerupere* este aproape sinonim cu *apel sistem* deoarece rutinele sistemului de operare şi BIOS-ului sunt apelate folosind instrucțiunea INT după pregătirea corespunzătoare a parametrilor în registre.

interrupt handler (IH) = rutină de tratare a întreruperii

Funcție (sau procedură, subrutină, modul program) asociat unei anumite întreruperi şi care se execută la apariția întreruperii pentru care a fost definită. Prin mecanisme specifice fiecărui sistem, la activarea ei se salvează starea programului întrerupt. După tratarea întreruperii în mod normal se reia programul întrerupt, dar, dacă prin întrerupere se realizează şi o replanificare pentru execuție, este posibil ca la revenire controlul să fie dat altui proces (la o altă adresă şi cu un alt context de execuție).

IRIX = IRIX

Versiunea de UNIX a Silicon Graphics Incorporated pentru propriile stații de lucru şi pentru servere mari de fişiere.

Iron Age = "epoca fierului" în construcția de calculatoare

În istoria calculatoarelor, sub acest nume este cunoscută perioada 1961-1971 în care s-au pus bazele producției şi comercializării de calculatoare. Memoriile bazate pe inele de ferită erau de dimensiuni importante, diferitele părți componente ale sistemului - unitatea centrală, memoria - fiind păstrate în dulapuri metalice impozante (de aici noțiunea de *mainframe*). Epoca fierului a început cu primul minicalculator (PDP-1) şi s-a încheiat cu introducerea primului microprocesor (Intel 4004).

Ironman = "Ironman"

Specificare a cerințelor pe care trebuie să le îndeplinească un limbaj de programare, întocmită la cererea DoD (US Department of Defense) în ianuarie 1977 şi revizuită în iulie 1977 în procesul de definire a mediului de programare Ada. Limbajul Ada a fost creat în urma unei activități interesante de evaluare a limbajelor de programare existente, încercându-se elaborarea unui standard al cerințelor la care ar trebui să răspundă un limbaj ideal. Au fost elaborate şi rafinate pe rând mai multe documente: Strawman, Woodenman, Tinman, Ironman, Steelman.

Litera J

job = lucrare

1. Ansamblu definit de prelucrări pe care un sistem de operare trebuia să le execute pentru un utilizator (de exemplu compilarea unui program, urmată de editarea de legături şi de două execuţii succesive cu două seturi diferite de date). Simultan cu specificarea activităţilor, puteau fi precizate şi resursele necesare execuţiei lor (mărimea partiţiei de memorie, timpul maxim de rulare, numărul de benzi magnetice solicitate, etc.), iar uneori se indica şi comportamentul aşteptat al lucrării (limitată unitate centrală, echilibrată, etc.). În contextul sistemelor interactive moderne, termenul îşi păstrează semnificaţia în legătură cu acţiuni pregătite şi planificate să se execute pe acelaşi sistem la un moment bine precizat, sau cu activităţi pregătite pe o maşină şi trimise spre a fi executate pe o altă maşină la distanţă.
2. (*cu caracter istoric*) Când utilizatorii aduceau la centrul de calcul programe perforate pe cartele, *lucrare (job)* era numele pachetului de cartele care conţinea o lucrare. Programul sursă şi datele de intrare erau însoţite de cartele de control care reprezentau comezile adresate de utilizator sistemului de operare. O primă cartelă servea la identificarea utilizatorului şi a lucrării, o ultimă cartelă marcând sfârşitul lucrării.

job control language (JCL) = limbaj de control al lucrărilor

1. Limbajul în care se descriu paşii diferitelor lucrări (comezi pe care utilizatorul le dă sistemului). Nu este vorba despre o simplă secvenţă de comenzi, pentru că în procesul de servire a cererilor pot apare decizii (nu se mai execută un pas dacă cel precedent s-a terminat cu eroare), sau cicluri (anumite acţiuni se repetă de un număr determinat de ori). El cuprinde atât instrucţiuni declarative cât şi imperative. Primele servesc la identificarea utilizatorului, a lucrării, a resurselor cerute, sau la stabilirea unui mediu de execuţie corespunzător. Cele din a doua categorie conduc efectiv la execuţia unor prelucrări cerute de utilizator. Deoarece noţiunea de lucrare este depăşită, dar cea de interfaţă cu utilizatorul îşi sporeşte importanţa, sistemele actuale utilizează astfel de limbaje, pe care le numesc simplu: *limbaje de control*.
2. La sistemele IBM OS/360, JCL era numele limbajului de control al execuţiei programelor în sistemul cu organizare a exploatării "pe loturi".

job queue = coada de lucrări

Mecanismul prin care lucrările sosite într-un sistem de tip "prelucrare pe loturi" (*batch processing*) aşteptau să fie luate în considerare de către planificatorul de lucrări. Deoarece sistemul nu dispunea de resurse nelimitate, o lucrare trebuia uneori să aştepte eliberarea unei partiţii de dimensiune corespunzătoare, a unei benzi magnetice alocate unei alte lucrări, etc. Servirea nu se făcea neapărat în ordinea sosirii (*fifo*), pentru că într-o partiţie liberă de 64Kb nu poate fi încărcată o lucrare care solicită 1Mb, chiar dacă ea este prima din coadă. În unele cazuri se organiza o coadă unică, în altele se menţineau mai multe cozi (câte una pentru fiecare partiţie).

job scheduling = planificarea lucrărilor

Decizia de a alege şi planifica pentru execuţie o nouă lucrare dintre cele aflate în aşteptare, atunci când sistemul are resurse disponibile (de exemplu prin terminarea unui program se creează o partiţie liberă). A fost o funcţie specifică sistemelor de operare cu prelucrare pe loturi, cunoscută şi sub numele de *"planificare pe termen mediu".* Se realiza înlănţuirea automată (fără intervenţia operatorului) a fazelor unei lucrări, precum şi a lucrărilor dintr-un lot.

job step = fază a unei lucrări (pas)

Paşii unei lucrări sunt acţiuni care se desfăşoară secvenţial. O compilare, o editare de legături, o execuţie a unui program pe un set de date de intrare sunt exemple tipice de astfel de paşi. Caracterul lor secvenţial provine din restricţii temporale: nu se poate lansa în execuţie un program până ce nu s-a terminat editarea de legături (care construieşte programul executabil), care la rândul ei nu poate începe până nu s-a terminat compilarea fără erori.

journal file = fişier jurnal

Fişier în care sistemul de operare consemnează evenimentele ce apar pe parcursul funcţionării sale.

Litera K

kernel = nucleu

Partea esenţială a unui sistem de operare, responsabilă cu gestiunea resurselor şi tratarea principalelor apeluri sistem pentru programele rulate deasupra nucleului. Tratarea întreruperilor, planificarea proceselor pentru execuţie (*planificarea pe termen*

scurt), sincronizarea proceselor, administrarea memoriei, accesul la disc şi la alte echipamente periferice, crearea de noi procese şi gestiunea acestora sunt câteva dintre funcţiile sale. Nucleul poate fi monolit (ca la UNIX) sau ierarhizat într-o structură multinivel. El poate avea o parte rezidentă şi una tranzitorie. În partea rezidentă se află concentrate funcţiile importante, frecvent utilizate şi care au prioritate mare în execuţie. În partea tranzitorie sunt încărcate şi executate la cerere module care reprezintă funcţii mai puţin importante sau cerute mai puţin frecvent. Când portabilitatea este principalul obiectiv al sistemului, nucleul poate fi ierarhizat într-o parte dependentă de maşină şi una independentă de arhitectură. Numai prima trebuie rescrisă la portarea sistemului de operare pe o altă structură de sistem.

kernel mode = mod sistem

Modul de lucru privilegiat al sistemului de calcul, în care se execută modulele program ale sistemului de operare. În acest mod se pot executa toate instrucţiunile (inclusiv cea de oprire a maşinii sau de schimbare a modului de lucru), se pot accesa toate resursele (registre de mapare pagini, cuvântul de stare program, măştile de priorităţi) şi se poate accesa orice zonă de memorie (inclusiv cea în care se află nucleul şi structurile sale de date, vectorii de întrerupere, etc.).

kill (to) = a distruge (un proces)

Metodă de terminare forţată a unui proces prin distrugerea acestuia, indiferent de starea în care se află. Se poate aplica şi altor module executabile gestionate de către sistem, de exemplu fire de execuţie ("threads"). Este necesară pentru a-i da utilizatorului posibilitatea de control asupra proceselor sale. Procesele pot bucla la nesfârşit, pot să evolueze în discordanţă cu cerinţele, se pot bloca în aşteptarea unor condiţii care nu se vor îndeplini niciodată sau pur şi simplu trebuie oprită execuţia unei aplicaţii care are mai multe procese.

Korn Shell (ksh) = ksh

Unul dintre interpretoarele de comenzi pentru UNIX.

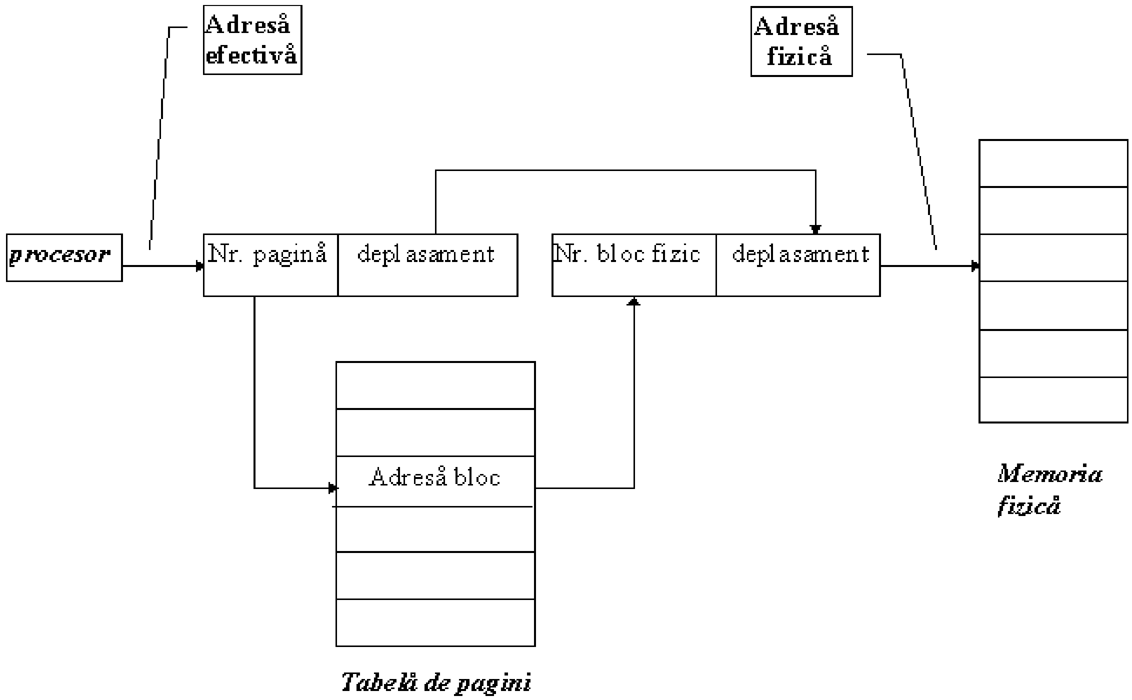
Litera P

page replacement algorithm = algoritm de înlocuire de pagină

În cazul administrării memoriei cu paginare la cerere, metodă utilizată de sistemul de operare pentru alegerea paginilor ce se vor scoate din memorie pentru a crea loc altor pagini ce trebuie încărcate în memorie. Teoretic, orice pagină se poate încărca în orice bloc (cu excepţia celor rezervate pentru sistemul de operare). Dacă pagina ce trebuie înlocuită a fost modificată, în timpul scurs de al ultima sa încărcare în memorie, ea va trebui rescrisă pe disc pentru a face actualizarea conţinutului. Dacă pagina nu a fost modificată, atunci nu mai trebuie rescrisă pe disc, deoarece copia sa se găseşte deja acolo. Printre cei mai utilizaţi algoritmi se numără: FIFO (First In First Out), LRU (Least Recently Used), MRU (Most Recently Used), precum şi algoritmi obţinuţi prin simplificarea celor de mai sus în scopul scăderii complexităţii, cum ar fi algoritmul ceasului, NUR (Not Used Recently), metoda dubleţilor etc. Performanţele acestor algoritmi se apreciază după numărul de adresări care necesită înlocuire de pagină.

page table = tabelă de pagini

Mulţime de registre sau structură de date de tip tablou conţinând informaţii necesare localizării paginilor în memoria fizică. Realizează corespondenţa între pagină şi adresa blocului de memorie fizică în care aceasta este încarcată la un moment dat (în cazul paginării la cerere această adresă se poate schimba în timp, pagina fiind înlocuită şi apoi reîncărcată la altă adresă fizică). Conţinutul registrului din tabelă corespunzător unei pagini participă la conversia adresei efective generate de procesor (formată din număr de pagină şi deplasament în pagină) în adresa fizică (formată din număr bloc memorie şi deplasament în bloc) după modelul prezentat în figura alăturată.



Obținerea adresei fizice din adresa efectivă

Arhitecturile de sistem de calcul scumpe conțin registre de pagină care asigură o viteză mare adresării unei pagini. Maşinile care acceptă spații mari de adrese sunt nevoite să păstreze tabelele de pagini în memorie şi să utilizeze o memorie asociativă rapidă sau alt mecanism care să limiteze scăderea vitezei datorită adresării suplimentare la memorie în vederea obținerii conținutului registrului de pagină. Pentru fiecare pagină, pe lângă adresă se mai păstrează şi alte informații: cheie de protecție pentru controlul accesului, înregistrarea gradului de utilizare a paginii în vederea aplicării corecte a algoritmilor de înlocuire de pagină, etc.

page-in = încărcare de pagină

1. Aducerea paginii (care conține adresa specificată în instrucțiunea executată de procesor) într-un bloc de memorie pentru a fi accesată. Dacă nu există un bloc liber, se utilizează un algoritm de înlocuire de pagină pentru a crea un astfel de spațiu liber. Operația de încărcare a paginii în memorie este o operație de intrare/ieşire care necesită un timp mare în raport cu timpul necesar execuției instrucțiunii. De aceea se utilizează memorii externe cu acces rapid şi ierarhii de memorii (memorii cache). Eventual, în timpul înlocuirii paginii se planifică pentru execuție un alt proces.
2. Sinonim cu swap-in (a se vedea "swapping").

page-out = eliminare de pagină

1. Transferul pe disc al unei pagini aflată într-un bloc de memorie, pagină aleasă în urma aplicării unui algoritm de înlocuire de pagină, pentru ca o nouă pagină să fie adusă în locul ei în memorie.
2. Sinonim cu swap-out (a se vedea "swapping").

paging = paginare

Tehnică de gestiune a memoriei prin care spațiul de adrese al programului este împărțit în unități de dimensiune fixă numite *pagini* care vor fi încărcate în *blocurile* în care este împărțită memoria fizică, practic paginile putând fi distribuite oriunde în memorie. În felul acesta se elimină fragmentarea memoriei şi necesitatea colectării spațiului liber. Paginile trebuiesc puse în corespondență ("mapate") cu blocurile pe care le ocupă, printr-un mecanism gestionat de unitatea de management al memoriei (MMU). În cazul în care nu se încarcă la un moment dat în memorie toate paginile programului, ci numai cele utilizate în acel moment, se realizează paginare la cerere, tehnică utilizată de majoritatea sistemelor de memorie virtuală; devine posibilă rularea unor programe de dimensiune mult mai mare decât memoria fizică disponibilă.

paging algorithm = algoritm de paginare

A se vedea "page replacement algorithm".

paging drum = tambur magnetic de paginare

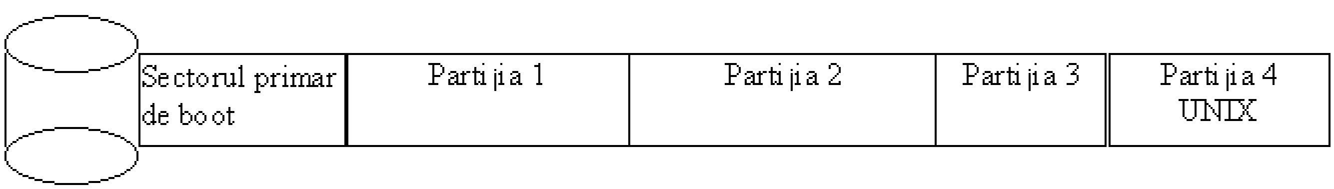
Echipament de intrare/ieşire de tip memorie auxiliară, din clasa dispozitivelor cu capete fixe, utilizat ca suport pentru mecanismul de paginare la cerere. Datorită timpului mic de acces şi a identităţii dimensiunii blocurilor sale cu dimensiunea paginii, a fost utilizat cu succes pentru evacuarea paginilor în primele sisteme de memorie virtuală. Are caracter istoric; asemenea dispozitive nu se mai produc în prezent.

parent process = proces "părinte"

Proces care a creat un alt proces (în relaţia cu procesul nou creat). În sistemul de operare UNIX, un proces poate crea un nou proces (proces "fiu") în urma apelului sistem **fork()**. Procesul nou creat este o copie identică a procesului "părinte" obţinută prin duplicarea tuturor zonelor de cod şi de date ale acestuia (cu excepţia zonei de cod pur '.text', partajată de cele două procese). În acest mod procesul "fiu" moşteneşte întregul context de execuţie al procesului "părinte", inclusiv fişierele deschise. Din acest moment se produce separarea, cele două procese având propriile imagini în memorie. Procesul "fiu" poate însă să decidă să ruleze un alt program (fişier executabil) utilizând apelul **exec().** Prin crearea de noi procese şi prin existenţa relaţiilor "părinte"-"fiu" se realizează o ierarhizare a proceselor, în sistem existând un arbore unic de procese. Procesul "părinte" se poate sincroniza cu procesele create de el: poate aştepta terminarea lor şi recupera informaţii emise de acestea cu această ocazie.

partition = partiţie

1. Zonă contiguă de memorie internă gestionată de către modulul de administrare a memoriei în cazul sistemelor de operare care realizează multiprogramare sau multiproces. În primele sisteme de operare cu multiprogramare o partiţie era alocată în întregime unei lucrări ("job"), în ea se rula un singur program, iar spaţiul neocupat de către acesta era pierdut (a se vedea "fragmentare internă"). Partiţionarea memoriei era statică (a se vedea "MFT"). Tehnica de partiţionare a evoluat spre specificarea dinamică a partiţiilor (a se vedea "MVT").
2. În sistemul de operare MS-DOS, secvenţă contiguă de sectoare logice ale unui disc, începând cu sectorul 0. Prin partiţionare se implementează mai multe volume logice pe un volum fizic. Se realizează astfel o mai bună protecţie între utilizatori, o gestiune mai eficientă a resurselor şi se oferă posibilitatea de a avea la dispoziţie mai multe sisteme de operare pentru a controla maşina în diferite momente. Un hard-disk poate fi împărţit în cel mult 4 partiţii din care numai una este activă la un moment dat (conţine sectorul de boot al sistemului de operare care va fi lansat în execuţie). Fiecare partiţie poate conţine propriul său sistem de fişiere, partiţii diferite putând avea sisteme de fişiere diferite (ceea ce permite ca DOS-ul şi Unix-ul să poată coexista pe acelaşi disc în partiţii diferite).



Exemplu de partiţionare a discului în PC

partition table=tabelă de partiţionare

În DOS, tabela aflată în ultima parte a sectorului de încărcare ce conţine toate informaţiile necesare identificării partiţiilor din sistem, stabilite prin FDISK.

password = parolă

Formă de autentificare a unui utilizator care presupune furnizarea unui cod (şir de caractere) pentru a obţine accesul la un sistem protejat sau la o resursă protejată. În momentul de faţă este cea mai frecventă metodă pentru controlul accesului utilizatorilor la un sistem, datorită costului redus, dar au apărut deja metode mai sigure şi mai sofisticate, dar mult mai scumpe (recunoaşterea vocii, cartelă magnetică furnizând o semnătură digitală, recunoaşterea scrisului prin analiză

grafologică, etc.). Aceste parole, stabilite şi modificate de către utilizator, sunt criptate după un algoritm de criptare specific fiecărui sistem şi memorate în această formă criptată pentru a face dificilă aflarea lor de către utilizatori rău intenționați. În UNIX parola de login a utilizatorului este memorată, în formă criptată, în fişierul *'/etc/passwd'* alături de: nume login, nume utilizator, identificator utilizator şi de grup, catalog implicit de lucru, shell-ul de login implicit. La fiecare cerere de acces, parola furnizată de utilizator este criptată, rezultatul criptării trebuind să coincidă cu cel memorat în sistem. În caz contrar se refuză accesul. Ca politică de securitate, unele sisteme dau posibilitatea ca administratorul de sistem să stabilească un timp limită de expirare a parolelor, aşa încât utilizatorul să trebuiască să-şi schimbe parola de login cu regularitate. Se recomandă utilizarea de parole greu de ghicit, cum ar fi cele ce conțin atât litere, cifre, cât şi simboluri speciale în diferite combinații (mulți spărgători utilizează termeni ai unor dicționare când forțează penetrarea unui sistem). A se vedea şi "Login".

path = cale

1. În UNIX şi alte sisteme de operare, prefix al unui nume de fişier care realizează specificarea completă relativ la catalogul rădăcină, indicând numele tuturor cataloagelor prin care se trece în ierarhia de fişiere până la catalogul care conține fişierul; în acest caz se vorbeşte despre 'cale absolută', spre deosebire de 'calea relativă' care face specificarea relativ la catalogul curent. Exemplu:

   ```
   /usr/prof/mdobre/dictionar/literap.txt
   ```

2. În UNIX şi MS-DOS există o 'cale de căutare', o variabilă de mediu numită PATH, care specifică directoarele în care interpretorul de comenzi (shell - UNIX sau command.com - MS-DOS) vor căuta comenzile. Elemente similare celor anterioare apar în UNIX şi în contextul unor aplicații sau medii de programare: de exemplu preprocesorul 'C' are o 'cale de căutare' pe care o utilizează la găsirea fişierelor de tip header utilizate de `#include`.
3. Adresa Internet de rutare explicită: o specificare 'nod-cu-nod' a legăturii dintre două maşini.

PC DOS = (sistemul de operare) PC-DOS

Sistem de operare dezvoltat pentru calculatoare personale (IBM şi compatibile construite pe baza unor procesoare din familia .x86). Nucleul şi diferitele componente se încarcă de pe disc dur sau flexibil. Numele este un acronim pentru **P**ersonal **C**omputer DOS (**D**isk **O**perating **S**ystem). Are trei componente principale conținute în fişierele:

- BIO.COM - destinat legăturii cu rutinele BIOS din memoria permanentă (PROM)
- DOS.COM - asigură legătura cu programul utilizator
- COMMAND.COM - prelucrează comenzile DOS.

Este un sistem monoutilizator, nu este multitasking, depinde de arhitectura hardware, iar sistemele de fişiere alcătuiesc o structură arborescentă (unică în fiecare partiție).

Perl = Perl

Limbaj de programare creat, dezvoltat şi întreținut de către Larry Wall. Este interpretat şi a fost dedicat pentru administrare de sistem, baze de date şi rețea UNIX. Are o răspândire deosebită în lumea administratorilor de sistem UNIX, tot aşa cum C este limbajul specific programatorilor de sistem UNIX. Un exemplu cunoscut de program cu largă răspândire scris în Perl este **`ftpmail`**.

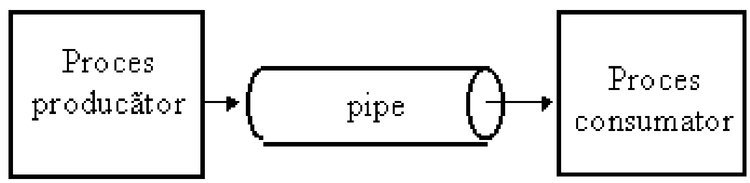
PDP-11 = PDP-11

Familie de calculatoare compatibile pe 16 biți construite de firma DEC. Au dominat lumea minicalculatorelor anilor 1970, fiind utilizate în special pentru aplicații de timp real, de control al proceselor industriale, dar şi dezvoltare de programe. Câteva exemple de maşini din familie: PDP 11/20, PDP 11/34, PDP 11/45, PDP 11/70. Maşinile aveau o arhitectură hardware uşor extensibilă: magistrală unică pe care se puteau conecta diferite echipamente. Ca mecanisme de protecție se ofereau cel puțin două moduri de lucru (*user* - mod în care rulează în mod curent un program, cu drepturi de acces limitate la nivelul fiecărui utilizator şi *kernel* - mod ce permite un acces la orice adresă, instrucțiune sau resursă), precum şi hardware de protecție a memoriei. Din punct de vedere al organizării memoriei, PDP-11 avea adrese virtuale pe 16 biți, până la 4 Mb de memorie fizică, pagini de memorie cu dimensiunea de 8Kb adresate prin 8 registre de pagină. Memoria era organizată astfel încât să permită acces atât la nivel de octet cât şi la cavânt. La proiectarea maşinii s-a optat pentru un compromis între o arhitectură orientată stivă şi una orientată spre registre generale: s-au prevăzut 8 registre generale (R0 - R5 pentru operanzi, dar şi posibili indicatori de vârf de stivă, R6 - indicator de vârf de stivă, R7 - contor program) şi 12 moduri de adresare, unele

dintre ele foarte puternice, făcând posibilă execuția unei operații *push* sau *pop* într-o singură instrucțiune. Lipsa instrucțiunilor de intrare/ieşire (registrele echipamentelor periferice fiind asimilate cu adrese de memorie dintr-o zonă protejată numită "pagina de intrare/ieşire") a favorizat compatibilitatea dintre maşinile din familie. Ca sisteme de operare s-au utilizat RSX-11M (sistem de timp real, multiuser şi multitasking, cu dezvoltări spre *time-sharing*) şi UNIX, pe aceste maşini fiind create şi distribuite multe dintre primele versiuni ale sistemului.

pipe = conductă

Mecanism de comunicare între procese ce implementează o relație de tip "producător-consumator", utilizând fişiere FIFO.



Procesele se consideră a fi conectate printr-un "canal" cu două extremități: una este extremitatea de scriere şi cealaltă de citire. Mecanismul are elemente comune cu comunicația între procese printr-un fişier obişnuit, dar prezintă două particularități importante:

- ordinea de manipulare a datelor este strict FIFO şi accesarea datelor se poate face o singură dată (First In First Out - primul venit primul iesit) în timp ce într-un fişier informația se poate accesa şi în altă ordine decât cea de scriere, aceeaşi informație putând fi accesată de mai multe ori.
- sincronizarea accesului proceselor la *pipe* este asigurată de către sistem în mod automat: procesul care încearcă să scrie într-un pipe plin va fi blocat până ce apare spațiu liber prin citirea de către alt proces a datelor din pipe; procesul care încearcă să citească dintr-un pipe vid va aştepta până când un alt proces va scrie ceva în pie. În funcție de atributele de deschidere a pipe-ului este posibil să nu se mai facă blocarea proceselor.

pool = ansamblu

Colecție de resurse echivalente (de exemplu zone tampon de intrare/ieşire), din care se face alocarea la cerere, urmând ca resursele eliberate să fie reintroduse în mulțime în vederea unei alocări ulterioare.

POSIX = POSIX

Proiect inițiat sub auspiciile Comitetului pentru Standarde al IEEE cu participarea unor personalități din industrie, lumea academică şi guvernul american cu scopul de a standardiza sistemul de operare UNIX. La sfârşitul anilor 1980 sistemele UNIX evoluaseră spre două variante care tindeau să devină total incompatibile (BSD şi System V). Fiecare având formate proprii pentru programele binare, producătorii de programe nu puteau realiza programe compatibile pentru ambele sisteme. Alte încercări de standardizare erau făcute de organizații care promovau interesele unor firme constructoare, ale căror produse deveniseră standarde de facto. POSIX este un acronim pentru **P**ortable **O**perating **S**ystem for UN**IX**. Printre standardele create de Comitetul POSIX se numără 1003.1 care defineşte un set de proceduri de bibliotecă pe care trebuie să le suporte orice sistem UNIX. Majoritatea acestor proceduri invocă un apel sistem, dar doar câteva dintre acestea pot fi implementate în afara nucleului. Ideea POSIX-ului este aceea că orice producător care scrie un program ce utilizează numai proceduri definite de 1003.1 este sigur că acel program va rula pe orice sistem UNIX. Pentru a crea acest standard, Comitetul IEEE a inclus în standard toate facilitățile şi apelurile comune ambelor versiuni (BSD şi System V). Toate componentele sistemului sunt avute în vedere în procesul de elaborare de standarde.

preemptive allocation = alocare cu prelevare forțată

Politică de alocare a resureselor dintr-un sistem care permite ca un proces să nu dețină o resursă pe toată perioada execuției

sale. Deoarece în sistem numărul de resurse fizice este limitat, se pune problema alocării de resurse astfel încât acestea să fie efectiv partajate de către procesele care le solicită. Această politică presupune că o resursă poate fi recuperată în orice moment de la procesul care o utilizează. Există resurse pentru care se poate face prelevare forțată (procesor, memoria internă, etc.) şi altele pentru care acest lucru nu este permis (bandă magnetică, imprimantă).

preemptive scheduling = planificare cu prelevare forțată

Politică de planificare a proceselor pentru execuție care permite ca un proces aflat în rulare să piardă resursa procesor la apariția unei cereri provenind de la un proces de prioritate mai mare, sau la expirarea cuantei de timp alocate. Procesul este trecut în starea "pregătit pentru execuție", un alt proces fiind planificat şi rulat conform algoritmului de planificare utilizat. În acest mod se realizează multiplexarea procesorului între procese.

priority = prioritate

Atribut care se asociază unui proces pentru a fi folosit la planificarea procesului pentru execuție atunci când se utlizează o politică de planificare bazată pe priorități. Prioritatea asignată la un moment dat unui proces poate fi dobândită static sau dinamic. Prioritatea statică este dată în momentul creerii procesului, în funcție de tipul de operații executate de acel proces (sau de clasa din care face parte procesul respectiv) şi rămâne nemodificată pe toată durata existenței procesului. Prioritatea dinamică este calculată periodic de către sistem în funcție de comportamentul procesului: ce procentaj din cuanta alocată a folosit, câte operații de intrare/ieşire a efectuat, ce masă de resurse deține deja, de când aşteaptă alocarea procesorului, cât timp a petrecut în memoria internă, etc. De regulă prioritățile sunt numere întregi care împart procesele într-un număr finit de clase.

priority queue = fir de aşteptare cu priorități

Structură de tip listă ordonată după prioritățile elementelor care o compun. Ordinea de servire nu este FIFO, ci una bazată pe prioritați. Cu toate acestea, pentru elementele cu aceeaşi prioritate, ordinea este FIFO. Poate fi considerată şi o listă de subliste care conțin elemente cu aceeaşi prioritate.

priority scheduling = planificare bazată pe priorități

Strategie de planificare pentru execuție a proceselor, care alege procesul cu prioritatea cea mai mare. În cazul existenței mai multor procese cu aceeaşi prioritate, se poate aplica o politică de servire FIFO, sau una de partajare a resurselor între procesele cu cea mai mare prioritate (round-robin pe sublista de prioritate maximă). Pentru a evita situația ca procesele cu prioritate maximă să ruleze la infinit, planificatorul de procese poate micşora prioritatea procesului care rulează după fiecare interval elementar de timp în care acesta a deținut procesorul. Dacă prioritatea procesului rulat devine mai mică decât cea a unui alt proces din sistem, procesul rulat este întrerupt, pus în coada asociată nivelului său de prioritate, iar noul proces cu prioritate maximă va fi rulat. Procesele care aşteaptă mult timp în subliste din care nu se face planificare pot fi trecute în subliste de prioritate mai mare pentru a li se da o şansă să utilizeze procesorul. În anumite sisteme de timp real se face prelevare forțată a procesorului la apariția unui proces prioritate mai mare, acesta primind procesorul pe timp nelimitat (până se termină, se blochează sau apare o cerere de prioritate mai mare ca a lui).

privileged account = cont privilegiat

Cont corespunzător unui utilizator cu privilegii deosebite în sistem. Unele sisteme de operare acceptă o clasă largă de astfel de utilizatori (de exemplu RSX-11M), în timp ce în UNIX se presupune de regulă existența unui singur utilizator privilegiat (superuser) având numele *root*. Acesta este contul din care se administrază sistemul. Lucrând în acest cont superuser-ul are toate drepturile în sistem, putând schimba permisiunile de acces la diferite resurse ale utilizatorilor obişnuiți, crea sau şterge utilizatori, permite sau interzice accesul în rețea, etc.

privileged instruction = instrucțiune privilegiată

Instrucțiune executată în modul de operare privilegiat (kernel) ce poate accesa toată memoria (atât zona user cât şi zona sistem), poate modifica sau vizualiza resurse protejate cum ar fi: cuvântul de stare program, registrele de pagină, listele de capabilități, măştile de priorități, intrările în vectorii de întrerupere, etc. În sistemele multiutilizator instrucțiunea de oprire a maşinii (*halt*) este o instrucțiune privilegiată.

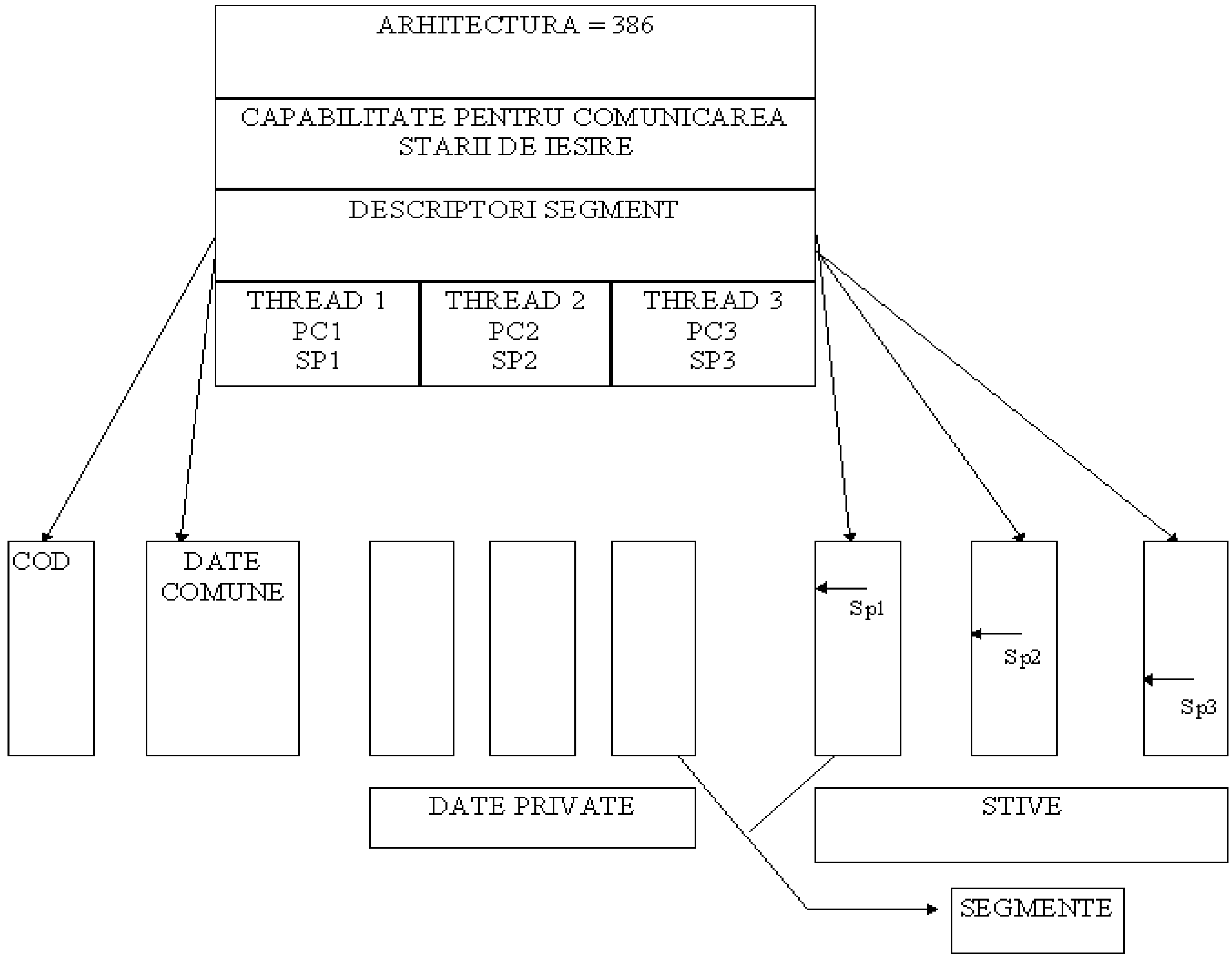
process = proces

Reprezintă o unitate de program executabilă gestionată de către sistemul de operare (este definit prin imaginea asociată constituită din codul care se execută, segmentul de date, de stivă, contorul program, conținutul registrelor şi în general toate informațiile care fac ca un program să poată fi rulat). În sistemele de operare mai vechi un proces era asociat unui program în execuție. În sistemele actuale un program poate fi format din mai multe procese care cooperează pentru rezolvarea problemei. Se accepta în trecut ideea că un proces este un program care se poate executa în paralel cu alte programe. În sistemele moderne un proces poate avea mai multe fire de execuție care se execută în paralel. Ele au propria lor stivă, dar partajează contextul de execuție al procesului din care fac parte.

process control block (PCB) = bloc de control al procesului



Structură de date utilizată de către sistemul de operare pentru a gestiona procesele existente în sistem. O astfel de structură conține informații cum ar fi: identificatorul procesului, starea curentă, prioritatea asociată, stiva, pointer la lista de procese care se află în aceeaşi stare. Un proces care face o cerere pentru o resursă temporar ocupată de alt proces (procesor, dispozitiv de intrare/ieşire, memorie, etc) este pus într-o listă de procese care aşteaptă la resursă prin inserarea PCB-ului său în această listă. La eliberarea resursei, planificatorul va parcurge o astfel de listă şi va alege noul proces care va utiliza resursa. În unele sisteme de operare există o tabelă de procese alocată static, fiecare intrare în tabelă fiind un PCB. În alte sisteme structura de date se alocă dinamic la creearea procesului şi se eliberează la terminarea sau distrugerea sa. A se vedea şi "TCB - task control block".



process descriptor = descriptor de proces

Structură de date asociată unui proces. În unele sisteme de operare sinonim cu bloc de control al procesului. Pentru sistemul distribuit Amoeba un descriptor are structura din figura alăturată. Un câmp din această structură conține tipul de arhitectură pe care poate rula procesul (într-o rețea eterogenă este esențial ca un proces în format binar petru '386 să nu ruleze pe un Sparc), un alt câmp conține o capabilitate pentru comunicarea stării de ieşire către procesor; când un proces se termină, procedurile de apel la distanță vor utiliza această capabilitate pentru a raporta evenimentele. De asemenea structura conține descriptori pentru toate segmentele proceselor, care definesc, în mod colectiv, spațiul de adrese. Există câte un descriptor pentru fiecare fir de execuție din proces (continutul acestui descriptor este dependent de arhitectură).

process identifier (PID) = identificator de proces

Identificator unic al unui proces într-un sistem de operare. Utilizat atât pentru gestiunea procesului de către sistem, cât şi ca adresă pentru comunicațiile între procese. În UNIX pid-ul unui proces are o valoare întreagă, unică la un moment dat în sistem. Fiecărui proces i se alocă o intrare din tabela de procese a sistemului, identificatorul de proces fiind clculat funcție de indicele intrării din tabelă corespunzătoare procesului în cauză.

process table = tabela de procese

Tablou (sau listă înlănțuită) de strucutri de date, una pentru fiecare proces existent în sistem. Fiecare intrare din tabelă conține o informație despre un proces, cum ar fi:

1. Parametrii de planificare: prioritatea procesului, timpul de procesor consumat, timpul cât procesul a fost în starea *dormant*. Acestea sunt utilizate în planificarea pentru execuție.
2. Imaginea memoriei: pointeri către segmentul de cod, date şi stivă sau, dacă se utilizează paginarea, pointeri la tabelele de pagini. Când procesul nu se află în memorie, aici se află informația despre regăsirea segmentelor pe disc.

3. Semnale: masca indicând semnalele ignorate, pe cele temporar blocate şi pe cele tratate în acel moment.
4. Diverse: starea procesului, evenimentul care este aşteptat să se producă, identificatorul propriu, identificatorul procesului părinte, identificatorul utilizatorului si al grupului.

Informația din tabelă nu este dusă niciodată pe disc; ea se află în permanență în memorie, pentru că sistemul gestionează şi procesele evacuate. În UNIX dimensiunea tabelei este fixă, dată de constanta NPROC; când tabela este plină nu se mai pot crea noi procese.

processor = procesor

Resursă foarte importantă, administrată cu prioritate maximă de orice sistem de operare concentrat sau distribuit. Procesorul unic sau procesoarele sunt alocate proceselor alese dintre cele pregătite pentru execuție, pe baza unei politici de alocare a procesoarelor implementată de planificatorul de procese.

processor allocation = alocare procesor

Algoritm utilizat pentru a decide care proces va fi rulat şi pe ce procesor în cazul unui sistem distribuit (care prin definiție se constituie ca o mulțime de procesoare ce pot fi organizate ca o colecție de stații de lucru, anamblu de procesoare sau organizare hibridă). Pentru modelul organizării sub formă de stații de lucru alocarea stabileşte când un proces rulează local sau când poate rula pe o altă stație neutilizată; pentru modelul ansamblului de procesoare, alocarea va decide pentru fiecare nou proces pe ce procesor se va rula.

processor time = timp procesor

Timpul în care procesorul este utilizat pentru execuția instrucțiunilor corespunzătoare proceselor din sistem.

producer = producător

(În contextul sistemelor de operare) proces care produce resurse temporare sub formă de mesaje care vor fi "consumate" de către un alt proces ("consumator"). Cele două procese pot fi conectate printr-o zonă tampon de dimensiune fixă. Între ele se stabileşte o relație specială: producător-consumator. Accesul la zona tampon poate fi sincronizat: din punctul de vedere al procesului producător aceasta înseamnă blocare în momentul în care încearcă să pună informația în tamponul plin.

profile = profil

1. Fişier de control pentru un program; se preferă fişiere de tip text pentru a permite modificarea uşoară de către utilizator. La lansarea programului este citit automat din catalogul de lucru al utilizatorului pentru a stabili comportamentul programului. Un exemplu este `.profile` din UNIX.
2. Raport despre durata de timp petrecută în fiecare rutină de către un program; este analizat pentru a studia comportamentul şi performanțele programului (un algoritm utilizat frecvent trebuie să aibă complexitate mică în timp ce unul utilizat foarte rar nu justifică uneori efortul de optimizare). Unele modele de profil raportează alte unități decât cele de timp (de exemplu numărul de apeluri) şi/sau fac măsurători la alte granularități decât per-rutină.

program relocation = relocare program

Tehnică de gestiune a memoriei care constă în deplasarea programului la alte adrese de memorie decât cele la care a fost încărcat inițial. Un program trebuie să fie construit astfel încât să permită relocarea (alternativa o constituie programele absolute, construite pentru a fi încărcate mereu la aceleaşi adrese de memorie). Fără relocare nu este posibilă nici multiprogramarea, nici multitaskingul.

program segmentation=segmentarea programului

Uneori dimensiunea unui program poate să depăşească memoria disponibilă pentru programele utilizatorilor sau spațiul de memorie adresabil de către program (cu registre de 16 biți acesta este limitat la 64Kb dacă nu se introduc mecanisme speciale). O tehnică menită să depăşească această limitare constă în divizarea înainte de execuție a spațiului de adrese al programului în zone ce reprezintă unități logice (funcții, structuri de date, tablouri, etc.) şi identificarea zonelor mutual exclusive. O astfel de zonă se încarcă în timpul execuției programului peste o altă zonă care nu este necesară în acel moment.

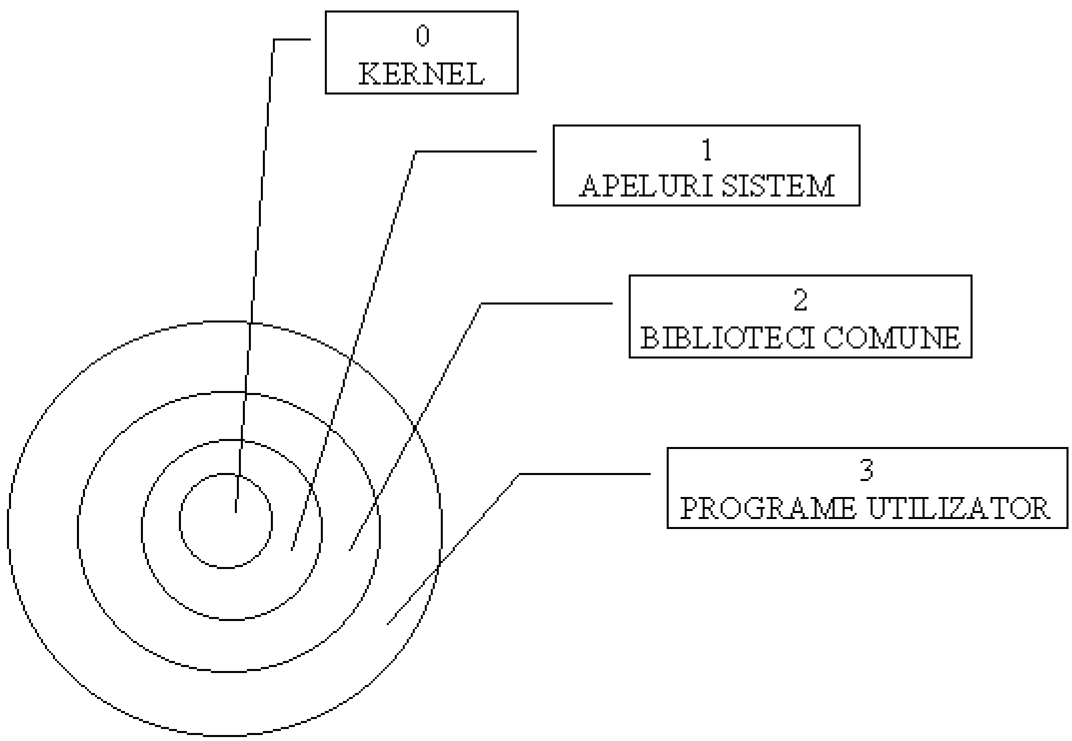
Se poate astfel executa un program mai mare decât memoria, dar se pierde timp cu transferul "segmentelor" de pe disc. Tehnica aceasta s-a numit inițial "segmentarea programelor", dar, după introducerea segmentării ca tehnică de memorie virtuală s-a preferat utilizarea termenului de "overlay". Sistemele mari oferă în general memorie virtuală, astfel încât tehnica descrisă se mai utilizează numai pe sisteme cu resurse sărace.

prompt = mesaj de invitație

Caracter sau şir de caractere pe care interpretorul de comenzi al unui sistem interactiv (shell-ul în UNIX) îl afişează pe ecran pentru a anunța utilizatorul că intrarea de la terminal va fi citită de către acest interpretor şi va fi considerată comandă (de la acelaşi terminal se poate interacționa şi cu alte programe). În Bourne-shell în mod implicit se utilizeză caracterul # pentru superuser şi $ pentru utilizatorul obişnuit. Utilizatorii pot alege alte mesaje de invitație cum ar fi: numele maşinii pe care lucrează la un moment dat, catalogul curent, etc. În mod normal, după citirea liniei de comandă se lansează în execuție comanda cerută (un program executabil). Acest program poate solicita intrare de la terminal sau nu. În momentul terminării comenzii, interrpretorul reafişează promptul, indicând utilizatorului că programul s-a terminat şi acum poate introduce o altă comandă.

protection ring = inel de protecție

Model de organizare ierarhică a protecției într-un sistem de operare introdus de MULTICS. Puterea unui utilizator (materializată prin drepturile sale de acces asupra diferitelor obiecte din sistem) depinde de inelul de protecție în care el evoluează. O exemplificare a acestui model este sugerată de figura alăturată. La nivel 0 lucrează nucleul sistemului de operare care gestionează memoria, procesele şi alte resurse. Apelurile sistem se execută la nivel 1. Accesul la bibilioteci comune implică o creştere a puterii (evoluție la nivelul 2) față de nivelul 3 la care rulează în mod normal programele utilizator. Acesta este nivelul cu cea mai slabă protecție. Creşterea puterii se face sub un control riguros al sistemului de protecție, după terminarea unui apel revenindu-se în inelul de putere mai mică de unde s-a pornit (un program care revine dintr-un apel sistem nu trebuie să păstreze drepturile de acces pe care le avea în timpul tratării apelului sistem.



purge (to) = a elimina versiunile vechi (ale fişierelor)

Operație de ştergere controlată a diferitelor versiuni ale fişierelor, în unele sisteme de operare. Unele sisteme de fişiere au posibilitatea să păstreze versiuni multiple ale fişierelor; o editare nu se termină cu pierderea vechii versiuni, ci cu crearea uneia noi, pe lângă cea veche. Aceasta poate să conducă la reducerea dramatică a spațiului disponibil pe disc, ceea ce implică eliminarea periodică a vechilor versiuni.

Litera Q

quantum = cuantă (tranşă de timp)

Interval de timp asignat în unele sisteme de operare unui proces căruia algoritmul de planificare i-a alocat procesorul. Este timpul maxim în care procesul poate utiliza procesorul până la o nouă planificare. Dacă procesul se blochează sau se termină înaintea expirării intervaluilui, se va face o planificare şi procesorul va rula alt proces. Dacă procesul mai rulează la sfârşitul cuantei de timp, procesorul este prelevat forţat şi alocat altui proces conform politicii de planificare. Alegerea cuantei este rezultatul unui compromis: cuante foarte mici cauzează multe comutări între procese şi determină o eficienţă scăzută a utilizării procesorului, în timp ce o cuantă prea lungă poate cauza un răspuns slab la cereri interactive.

queue = coadă (fir de aşteptare)

Structură de date sub formă de tablou static sau listă înlănţuită în care disciplina folosită pentru adăugarea şi eliminarea informaţiei asociate fiecărui element din coadă este strict FIFO.

queue discipline = disciplina asociată cozii

Politică utilizată într-o structură de tip coadă pentru gestionarea informaţiei adăugate sau extrase din ea.

queue management = gestiunea cozii

Totalitatea operaţiilor de introducere sau extragere a informaţiilor din coadă, respectându-se disciplina asociată cozii.

Litera R

race condition = condiţie de cursă

Situaţie în care datorită faptului că două sau mai multe procese utilizează date comune rezultatul final depinde de ordinea în care rulează fiecare proces şi momentele în care sunt modificate şi citite valorile variabilelor comune. Variabilele comune pot reprezenta structuri de date asociate unor resurse fizice utilizate în comun.